Equivalência de Fluxos e Modelagem Hierárquica

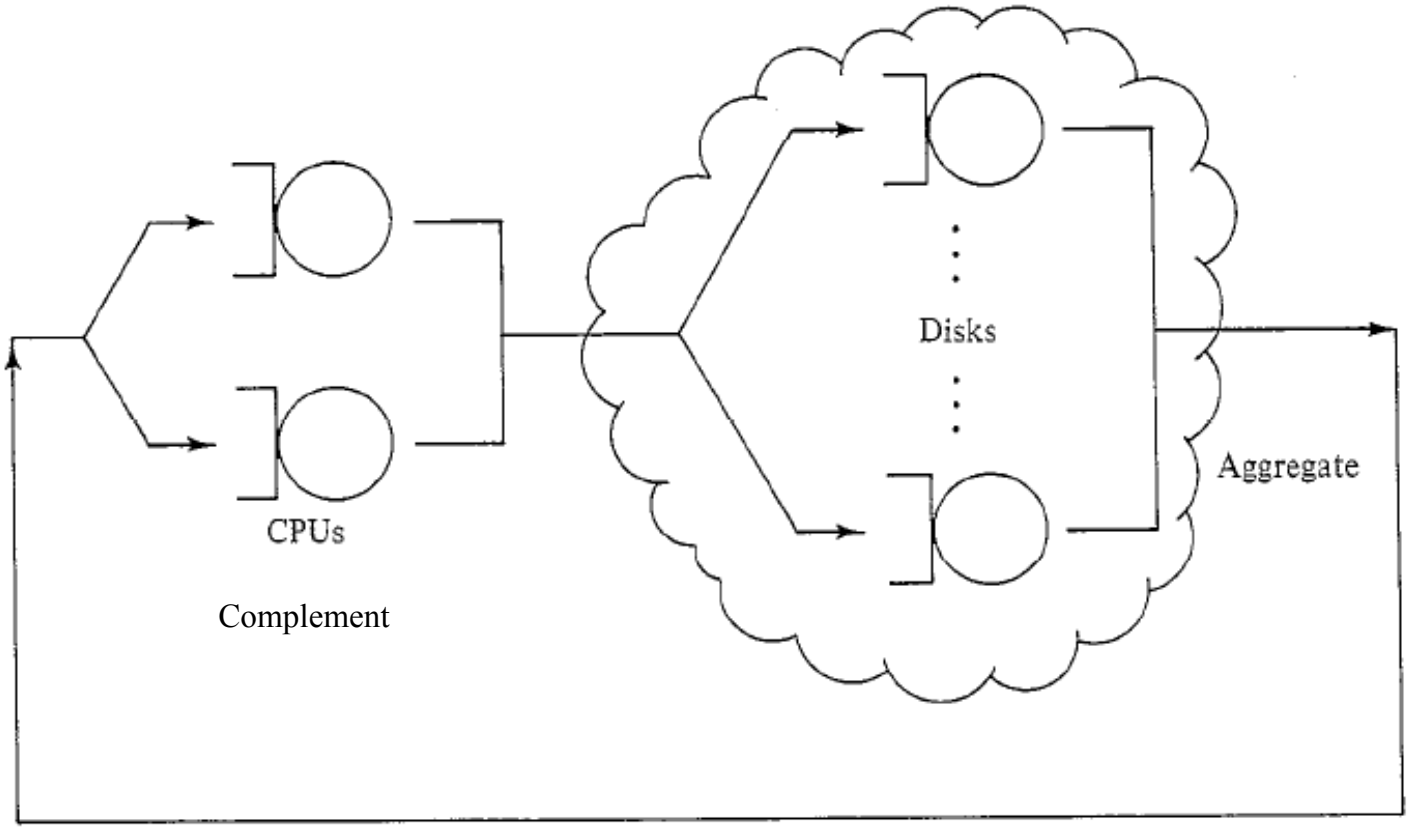
Profa. Jussara M. Almeida

1º Semestre de 2014

Modelagem Hierárquica

- Modelos mais sofisticados que podem incluir detalhes adicionais do sistema sendo representado
- Partição de um modelo grande em um número de submodelos menores
- Cada submodelo é avaliado (p.ex: MVA)
- Soluções individuais são combinadas para obter solução do modelo original
- Combinação feita através de um tipo especial de centro de serviço
  - FESC: flow equivalent service center

Modelagem Hierárquica: Exemplo



Um FESC é um único centro de serviço que, do ponto de vista da rede complementar, se comporta *identicamente* ao agregado

(causa mesmo atraso médio: mesmo tempo de residência médio e throughput)

Um FESC é uma aproximação (médias iguais mas distribuição pode ser diferente)

Modelagem Hierárquica: Exemplo

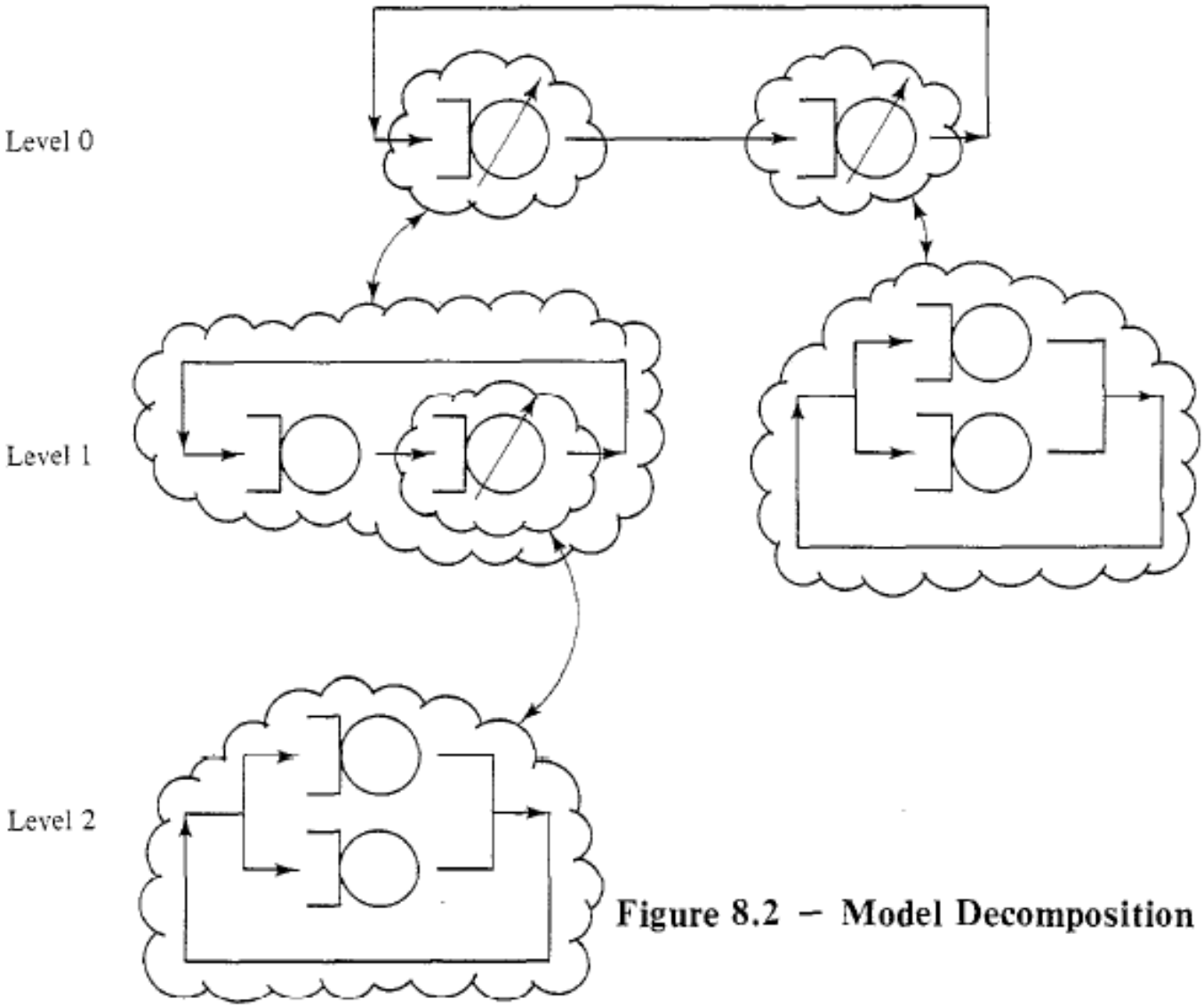


Figure 8.2 – Model Decomposition

Modelagem Hierárquica

- Embora a definição do modelo normalmente proceda do mais alto nível para o mais baixo nível, a avaliação dos mesmos ocorre na direção oposta
- Decomposability assumption: a taxa média com que clientes partem do agregado depende somente o *estado* do agregado, onde o *estado* é definido pelo número de clientes dentro do mesmo
  - Estado é independente da localização dos clientes nos vários centros de serviço que compõem o agregado.
- Aproximação provida pelo FESC é boa quando as taxas de serviço dentro do agregado são bem mais rápidas que as taxas dos centros no complemento.
  - Atinge equilíbrio local dentro do FESC
  - Se não atingir equilíbrio: o throughput do agregado pode depender da localização dos clientes nos centros que o compõem

Modelagem Hierárquica

- FESC = **load dependent service center**
  - Taxas de serviço $\mu(N) = X(N)$
    - throughput do agregado para todas as populações N
    - resolve agregado (nível l+1) e usa throughput como parâmetro para solucionar complemento (nível l)
      - Solução do nível l+1: experimentação, simulação, modelagem analítica (geral ou específica)
    - para cargas abertas (população ilimitada), computar X(N) para todos valores de N até certo limite prático
  - Soluções para modelos com classe única e com múltiplas classes
  - Lazowska, caps 8 (9-11), cap 20
  - Virgílio , cap 14.

Load Dependent Service Center

- Algoritmo MVA: revisão dos passos principais
  1. Calcule os tempos de residência (para cada classe) em cada centro, baseado nas demandas (da classe) e no número médio de clientes vistos quando da chegada de um novo cliente (da classe) ao centro
  2. Calcule o throughput (de cada classe) usando Lei de Little
  3. Calcule os tamanhos médios da fila em cada centro (para cada classe) usando Lei de Little ($Q = XR$)
- Este algoritmo não funciona para centros de serviço load dependent
  - Premissa de homogeneidade de tempo de serviço é violada

Load Dependent Service Center

- Passos 1 e 3 do algoritmo precisam ser modificados
  1. Taxas de serviço variam com tamanho da fila, logo o cálculo dos tempos de residência devem ser estendidos para refletir a variação nas filas e nas taxas de serviços

$$R_k(n) = V_k \sum_{j=1}^{n} \frac{j}{\mu_k(j)} p_k(j-1 \mid n-1)$$

onde $p_k(j \mid n)$ é a proporção do tempo em que centro k tem j clientes quando a população é igual a n

$\mu_k(j)$ é a taxa de serviço do centro k quando há j clientes

$p_k(j\text{-}1 \mid n\text{-}1)$ = Prob (um cliente chegando vê j-1 clientes no centro k dado que há n clientes no sistema) [Teorema da Chegada]

Load Dependent Service Center

- Passos 1 e 3 do algoritmo precisam ser modificados
  1. Taxas de serviço variam com tamanho da fila, logo o cálculo dos tempos de residência devem ser estendidos para refletir a variação nas filas e nas taxas de serviços

$$R_k(n) = V_k \sum_{j=1}^{n} \frac{j}{\mu_k(j)} p_k(j-1 \mid n-1)$$

Sejam $\alpha_k(j) \equiv \frac{\mu_k(j)}{\mu_k(1)}$ e $S_k = \frac{1}{\mu_k(1)}$ o tempo de serviço médio

quando há um cliente na fila, então:

$$R_k(n) = \frac{V_k}{\mu_k(1)} \sum_{j=1}^{n} \frac{j}{\alpha_k(j)} p_k(j-1 \mid n-1)$$

$$R_k(n) = D_k \sum_{j=1}^{n} \frac{j}{\alpha_k(j)} p_k(j-1 \mid n-1)$$

Load Dependent Service Center

- Passos 1 e 3 do algoritmo precisam ser modificados

  3. Nos centros *load independent*, somente os tamanhos médios de fila são calculados. No caso de centros *load dependent*, é preciso determinar a distribuição dos tamanhos da fila, isto é a proporção do tempo em que cada tamanho possível de população existiu no centro.

$$p_k(j \mid n) = \begin{cases} \dfrac{X(n)}{\mu_k(j)}\, p_k(j-1 \mid n-1) & j = 1, \ldots, n \\ 1 - \displaystyle\sum_{i=1}^{n} p_k(i \mid n) & j = 0 \end{cases}$$

Sabendo que $p_k(0 \mid 0) = 1$, para todo centro k

Lembre-se que: $\dfrac{1}{\mu_k(j)} = \dfrac{D_k}{\alpha_k(j)}$ pois $D_k = \dfrac{1}{\mu_k(1)}$

Load Dependent Service Center

- Passos 1 e 3 do algoritmo precisam ser modificados
  3. Nos centros *load independent*, somente os tamanhos médios de fila são calculados. No caso de centros *load dependent*, é preciso determinar a distribuição dos tamanhos da fila, isto é a proporção do tempo em que cada tamanho possível de população existiu no centro.

$$p_k(j \mid n) = \begin{cases} \dfrac{D_k X(n)}{\alpha_k(j)} p_k(j-1 \mid n-1) & \text{j} = 1, \ldots \text{n} \\ 1 - \displaystyle\sum_{i=1}^{n} p_k(i \mid n) & \text{j} = 0 \end{cases}$$

Sabendo que $p_k(0 \mid 0) = 1$, para todo centro k

Lembre-se que: $\dfrac{1}{\mu_k(j)} = \dfrac{D_k}{\alpha_k(j)}$ pois $\mathrm{D_k} = \dfrac{1}{\mu_k(1)}$

Algoritmo MVA com uma classe e centros LD

- Parâmetros de entrada:
  - D_k, N, K, e multiplicadores das taxas de serviço $\alpha_k(j)$

- Inicialização:

  for k = 1 to K do $p_k(0 \mid 0) = 1$ e $Q_k(0) = 0$

Algoritmo MVA com uma classe e centros LD

for n = 1 to N do

for k = 1 to K do

$$R_k = \begin{cases} D_k(1 + Q_k(n-1)) & \text{(centros LI)} \\ D_k & \text{(centros de atraso)} \\ D_k \sum_{j=1}^{n} \frac{j}{\alpha_k(j)} P_k(j-1 \mid n-1) & \text{(centros LD)} \end{cases}$$

$$X(n) = \frac{n}{\sum_{k=1}^{K} R_k}$$

for k = 1 to K do

$Q_k = XR_k$ (centros LI ou de atraso)

for j = 1 to n do

$$P_k(j \mid n) = \frac{D_k}{\alpha_k(j)} XP_k(j-1 \mid n-1) \quad \text{(centros LD)}$$

$$P_k(0 \mid n) = 1 - \sum_{j=1}^{n} P_k(j \mid n) \quad \text{(centros LD)}$$

$$Q_k = \sum_{j=1}^{n} jP_k(j \mid n) \quad \text{(centros LD)}$$

Modelagem Hierárquica (Modelos Separáveis)

Given a closed, separable model with K centers and population $\vec{N}$, let centers 1 through A represent the aggregate, and centers $A+1$ through K the complement.

. Create a low-level model by setting the service demands of centers $A+1$ through K to zero for all classes. This is equivalent to creating a model with centers 1 through A.

2. Evaluate this (separable) model with population $\vec{N}$, using the exact MVA solution technique. Obtain system throughputs $X_c(\vec{n})$ for all classes c and all populations from no customers to the full population $\vec{N}$.

3. Create a high-level model consisting of centers $A+1$ through K, an FESC representing centers 1 through A, and customer population $\vec{N}$. The service rate of the FESC for class c when the customer population in its queue is $\vec{n}$ should be $X_c(\vec{n})$.

4. Evaluate this high-level model using the extension to MVA described in Chapter 20. The solution of this model is an approximation to the solution of the original K center network. System performance measures for all customer classes, and performance measures for centers $A+1$ through K, are obtained as the results of this solution. Performance measures for centers 1 through A can be computed by combining information from the solutions of the high- and low-level models.

Exemplo 1

Virgílio, cap 14

Um servidor web (software Apache) tem dois processadores e 1 disco. Benchmarks indicam que um servidor com 2 processadores é 1.8 vezes mais rapido que um servidor com 1 processador para este tipo de carga. Logo:

$$\alpha_{processor}(j) = \begin{cases} 1 & j = 1 \\ 1.8 & j \geq 2 \end{cases}$$

As demandas por serviço de uma requisição HTTP no disco e no processador são 0.06 s e 0.1 s, respectivamente. Por simplicidade, assuma que o número máximo de threads abertas no Apache seja 3. Qual o tempo de resposta médio e o throughput do servidor?

Exemplo 1

Utilizando o algoritmo anterior, são obtidos os valores:

$$U_{processor}(n) = \sum_{j=1}^{n} p(j \mid n) = 1 - p(0 \mid n)$$

n = 0 : $Q_{processor} = 0$, $Q_{disk} = 0$, $p_{processor}(0|0) = 1$

n = 1: $R_{processor} = 0.1$, $R_{disk} = 0.06$, $R = 0.16$

$X = 6.25$, $Q_{processor} = 0.63$, $Q_{disk} = 0.37$

$p_{processor}(0 \mid 1) = 0.375$ $p_{processor}(1|1) = 0.625$

$U_{processor} = 0.625$, $U_{disk} = 0.375$

n = 2: $R_{processor} = 0.11$, $R_{disk} = 0.08$ $R = 0.19$

$X = 10.56$, $Q_{processor} = 1.13$, $Q_{disk} = 0.87$

$p_{processor}(0 \mid 2) = 0.238$ $p_{processor}(1|2) = 0.396$ $p_{processor}(2|2) = 0.367$

$U_{processor} = 0.763$, $U_{disk} = 0.633$

n = 3: $R_{processor} = 0.13$, $R_{disk} = 0.11$ $R = 0.24$

$X = 12.5$, $Q_{processor} = 1.60$, $Q_{disk} = 1.375$

$p_{processor}(0|3) = 0.173$ $p_{processor}(1|3) = 0.297$

$p_{processor}(2|3) = 0.275$ $p_{processor}(3|3) = 0.255$

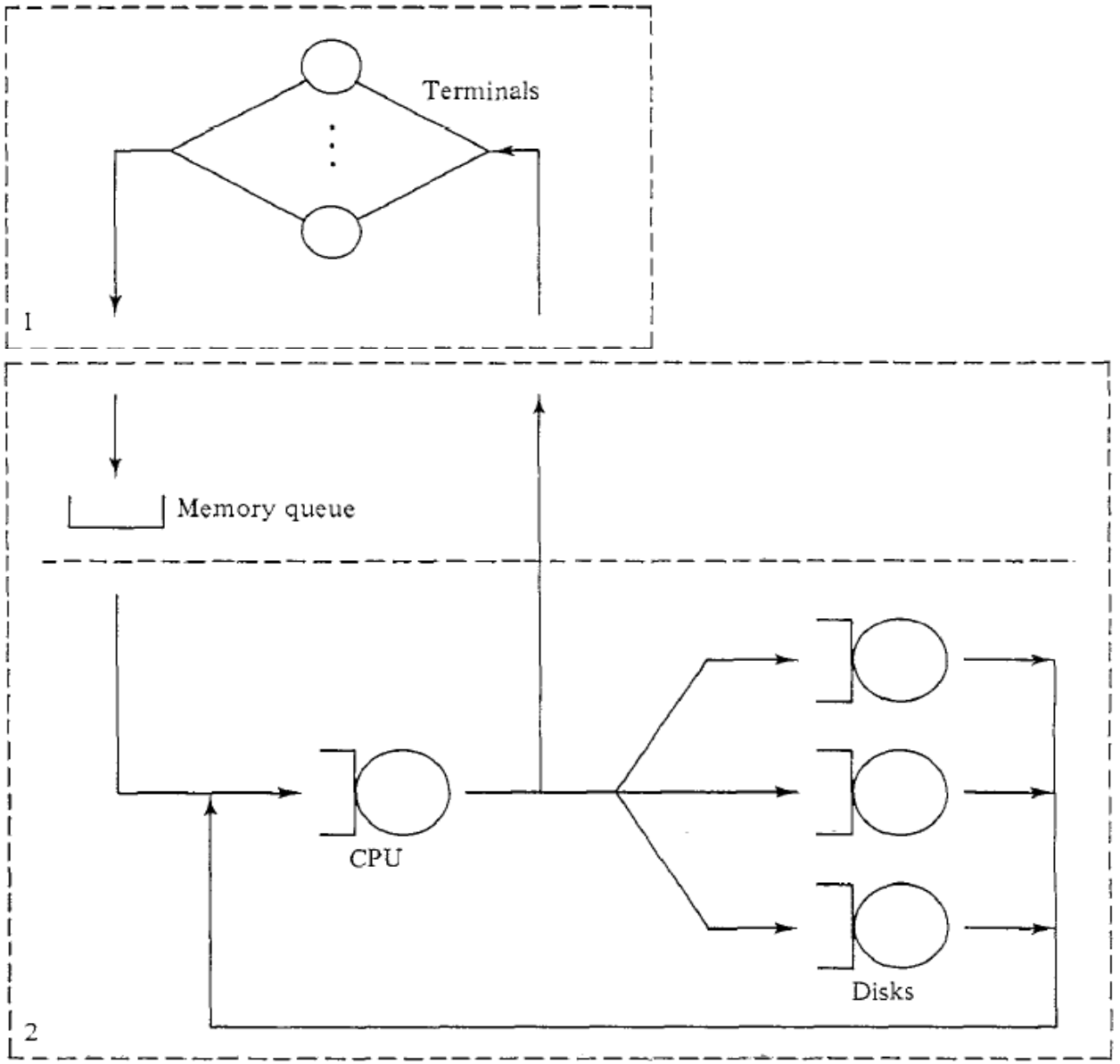
$U_{processor} = 0.823$, $U_{disk} = 0.747$

Restrição de Memória

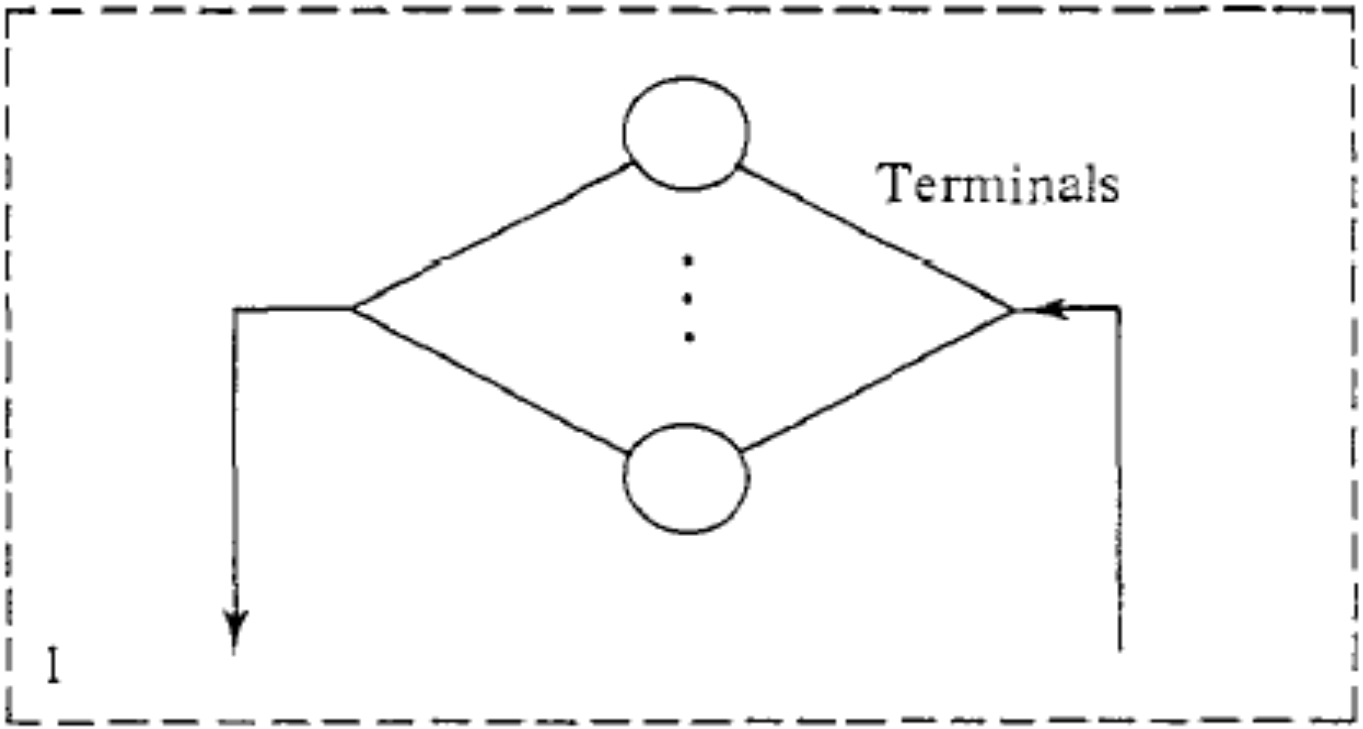
Lazowska, cap 9

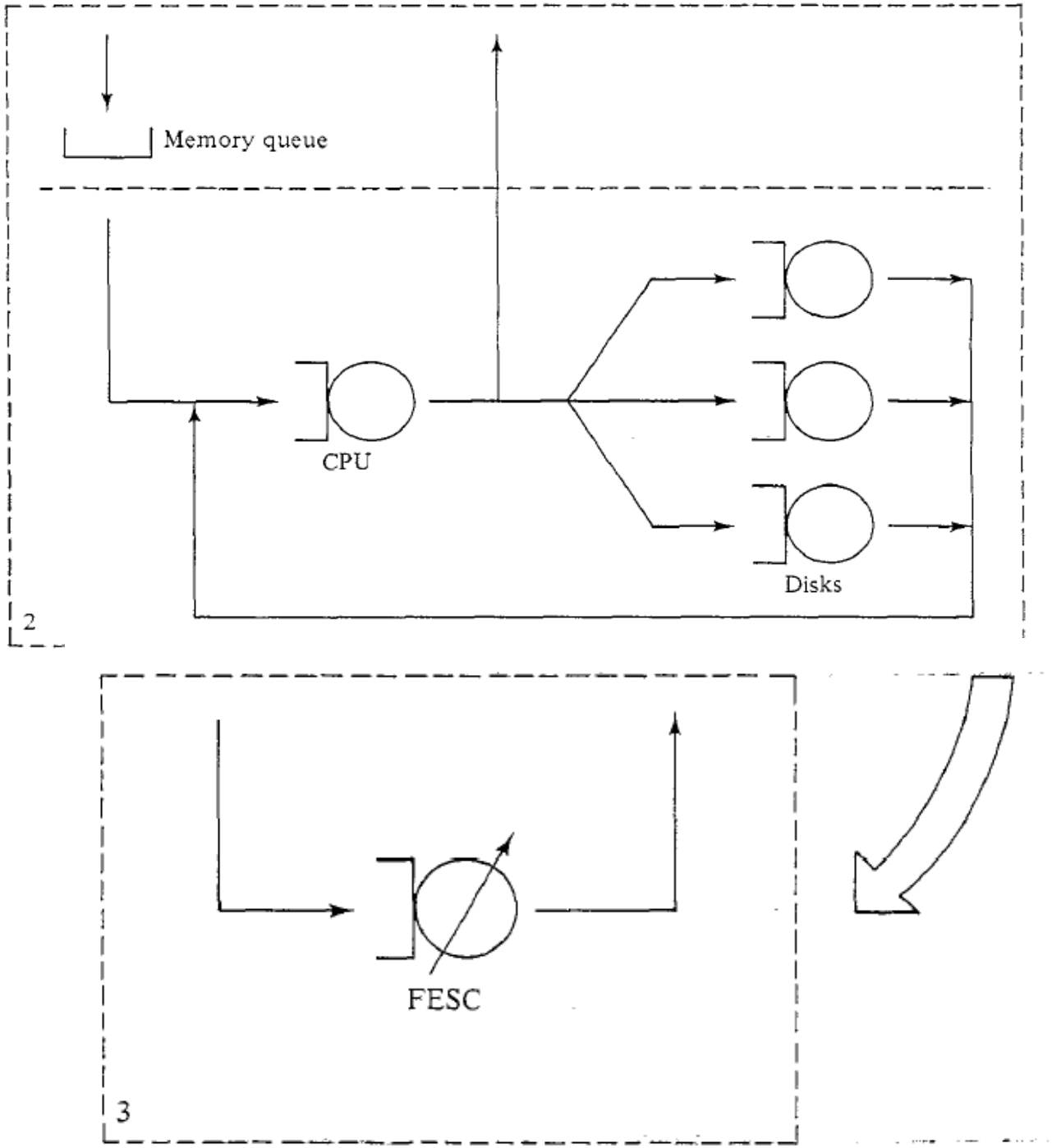
- Principal impacto da restrição de memória: throughput
  - Overhead de gerenciamento de memória: impacto secundário
- Note que casos extremos já são capturados:
  - Restrição de memória sempre alcançada: modelos batch
  - Restrição de memória nunca alcançada: modelos abertos e interativos
- Caso interessante: restrição de memória é às vezes (mas não continuamente) alcançada
  - Viola condição de separabilidade
  - Soluções MVA não são válidas
- Abordagem: modelagem hierárquica

Restrição de Memória



Restrição de Memória




Memory queue
CPU
Disks
2
FESC
3

Motivação: Exemplo

quantity		period 1	period 2	period 3	average
time interval		0 - 1268	1268 - 1734	1734 - 2108	
duration		1268	466	374	
proportion of total		.602	.221	.177	
MPL	workload 1	2	2	3	2.18
	workload 2	1	0	0	0.60
	workload 3	2	3	0	1.87

Table 9.1 – Variation in Multiprogramming Level (MPL)

Motivação: Exemplo

quantity		period 1	period 2	period 3	average
CPU utilization		.925	.782	.557	.825
throughput, jobs/minute	wkld. 1	1.343	1.475	2.498	1.58
	wkld. 2	14.71	0	0	8.86
	wkld. 3	29.71	44.14	0	27.6

Table 9.3 – Model Outputs for Three Time Periods

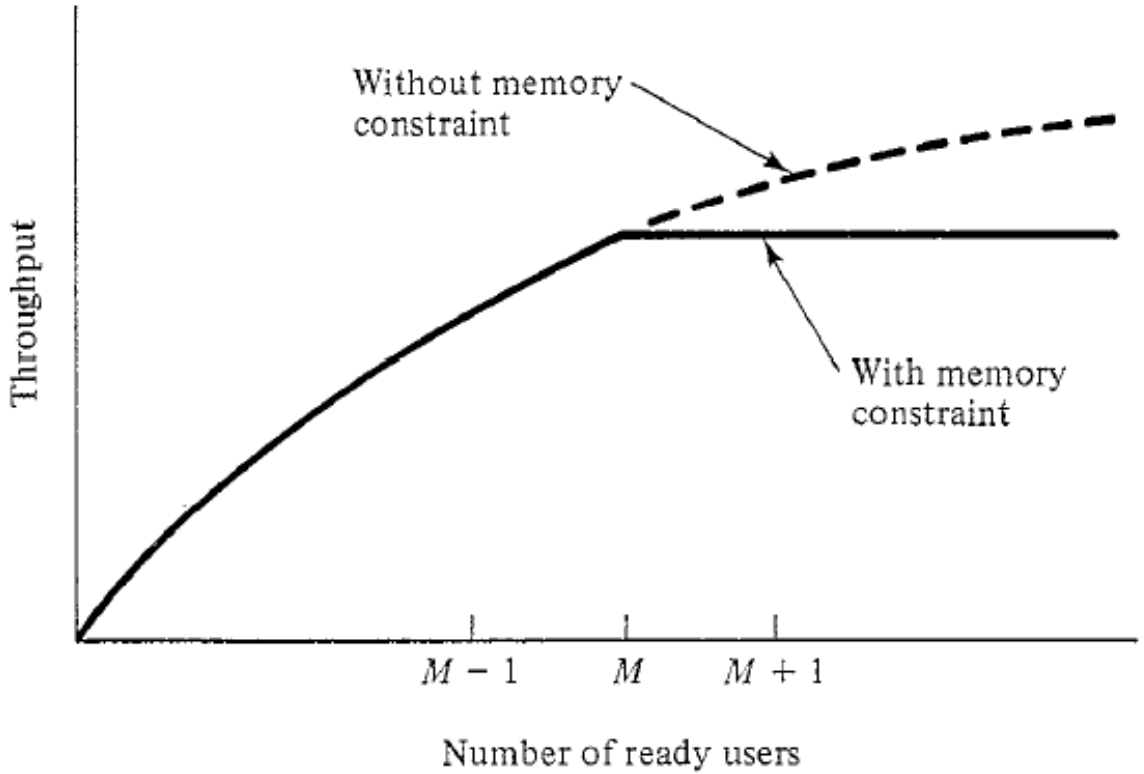
Motivação: Exemplo

quantity		actual value	model results			
			average MPL		variable MPL	
			value	discrep.	value	discrep.
CPU utilization		.820	.819	0	.825	+ 1%
t'put., jobs/min.	wkld. 1	1.59	1.51	− 5%	1.58	− 1%
	wkld. 2	8.77	8.72	− 1%	8.86	+ 1%
	wkld. 3	27.0	28.9	+ 7%	27.6	+ 2%

Table 9.4 − Measurements Versus Two Modelling Approaches

Impacto da Restrição de Memória no Throughput

- Modelo com uma única classe : requisitos de memória homogêneos
- Restrição de memória: M clientes simultâneos
  - Se um cliente fica pronto pra processar quando há M ou mais clientes prontos, então aquele cliente precisa esperar por memória



Impacto da Restrição de Memória no Throughput

FESC queue length	ready customers	active customers	throughput
1	1	1	$X(1)$
2	2	2	$X(2)$
⋮	⋮	⋮	⋮
$M-1$	$M-1$	$M-1$	$X(M-1)$
M	M	M	$X(M)$
$M+1$	$M+1$	M	$X(M)$
⋮	⋮	⋮	⋮
N	N	M	$X(M)$

Algoritmo

1. Define a low-level model consisting of the service centers representing the processing resources that comprise the central subsystem.
2. Evaluate this model, which is separable, for each feasible population, $n = 1, \ldots, M$. Note the load dependent throughputs, $X(n)$.
3. Create a load dependent service center that is flow equivalent to the central subsystem plus the memory queue, by setting its throughput with queue length n, $\mu(n)$, to:

$$\mu(n) = \begin{cases} X(n) & n = 1, \ldots, M \\ X(M) & n > M \end{cases}$$

4. Define a high-level model consisting of this FESC and the external environment: if a terminal workload, then N customers with think time Z; if a transaction workload, then an external arrival rate λ. Evaluate this model, which is separable.

Exemplo 2

Lazowska, cap 9

Considere um sistema com 1 CPU, dois discos e 1 GB de memória. Uma interação média requer 3 segundos de CPU, 4 segundos em um dos discos e 2 segundos no outro. Além disto, sabe-se que o consumo de memória pelo sistema operacional é de 50 MB e a demanda média por memória para cada interação é de 250 MB. Sabe-se que 15 usuários compartilham o sistema, com think time médio de 60 segundos. Quais são:

- o tempo de resposta médio?
- os números médios de clientes prontos e ativos?
- a distribuição do número de clientes ativos (em memória)?
- o tempo médio gasto esperando por memória para executar?
- a utilização de cada recurso de processamento (CPU e disco)?
- a melhoria no tempo de resposta médio se a quantidade de memória aumentasse de 600 MB?

Exemplo 2

Lazowska, cap 9

Se cada interação requer 250 MB de RAM, em média, o SO requer 50 MB e tem 1GB disponível:

no máximo 3 clientes podem estar residentes em memória simultaneamente (M = 3)

Passo 1: analise o sistema central (CPU e dois discos) para n = 1..3

MVA com K = 3, D_{cpu} = 3, D_{disk1} = 4, D_{disk2} = 2

population	throughput, interactions/sec. $= \mu(n)$
1	0.1111
2	0.1636
3	0.1930

Exemplo 2

Lazowska, cap 9

Passo 2: defina um modelo de mais alto nível com:

N=15 clientes, Z = 60 segundos

um centro LD que é FESC ao subsistema central mais a fila de memória

Neste caso, temos o throughput do FESC para todas as populações possíveis

queue length	throughput
1	0.1111
2	0.1636
3	0.1930
4	0.1930
⋮	⋮
15	0.1930

$= \mu(n)$

Exemplo 2

Lazowska, cap 9

Avalie o modelo de alto nível com o FESC, obtendo:

throughput: 0.175 interactions/second
average residence time at the FESC: 25.7 seconds (Tempo de resposta do sistema)
average queue length at the FESC: 4.5 (Número médio de clientes prontos: N=XR)
queue length distribution at the FESC:

queue length	probability
0	.038
1	.086
2	.122
3	.137
4	.142
5	.135
6	.117
>6	.228

Ocupação de memória por clientes:

3.8% do tempo: n = 0

8.6% do tempo: n = 1

12.2 % do tempo: n =2

75.4% do tempo: n ≥3 (Q_{FESC} ≥3)

Número médio de clientes ativos:

N_{ativos} = 0.086*1 + 0.122*2 +
0.754*3 = 2.6

Exemplo 2

Lazowska, cap 9

Avalie o modelo de alto nível com o FESC, obtendo:

throughput: 0.175 interactions/second
average residence time at the FESC: 25.7 seconds
average queue length at the FESC: 4.5
queue length distribution at the FESC:

queue length	probability
0	.038
1	.086
2	.122
3	.137
4	.142
5	.135
6	.117
>6	.228

Tempo médio gasto no subsistema central (Little):

$R = N / X = 2.6/0.175 = 14.9$ s

Tempo médio esperando por memória

$W = 25.7 - 14.9 = 10.8$ s

Utilização dos dispositivos ($U_k = XD_k$)

$U_{cpu} = 0.175*3 = 52.5\%$

$U_{disk1} = 0.175*4 = 70\%$

$U_{disk2} = 0.175*2 = 35\%$

Exemplo 2

Lazowska, cap 9

Impacto da memória adicional: 1.6GB -> ≤ 6 clientes simultâneos

Reavalie taxas de serviço do FESC para n = 4, 5 e 6

queue length	throughput
1	0.1111
2	0.1636
3	0.1930
4	0.2110
5	0.2226
6	0.2305
7	0.2305
⋮	⋮
15	0.2305

Reavalie modelo de mais alto nível com novo FESC, obtendo:

R = 20.7 segundos, uma melhora de 20%

Exemplo 2

Lazowska, cap 9

A utilidade do algoritmo proposto para tratar restrições de memória vem da sua precisão e da sua eficiência:

- precisão: a premissa de decomposability vale para os terminais e subsistema central uma vez que a taxa com que clientes interagem no subsistema central é muito maior que a taxa com que eles fluem entre os estados de thinking (terminais) e ready (subsistema central)
- Eficiência: tanto o modelo de mais baixo nível (necessário para se obter os throughputs em função da carga) quanto o de mais alto nível (com o FESC) podem ser analisados eficientemente. São modelos de uma única classe e separáveis

Mais além ....

- Overhead de swapping (algoritmo iterativo: Lasowska, cap 9.4)
  - Tratar swapping device como um centro de serviço
  - Precisa estimar tempo gasto para realizar fazer swapping (in and out): S_{swap}
  - Precisa estimar prob. de swapping P_{swap}
    - Depende de população N, restrição de memória M e

      # médio de clientes prontos Q_{ready} (= Q_{FESC})
  - Assim tem-se $D_{swap} = S_{swap} * P_{swap}$
    - diferentes estratégias para estimar P_{swap}
- Subsistema de I/O e processadores (incluindo diferentes políticas de escalonamento)
  - Modelagem hierárquica: vide Lasowska, caps 10 e 11

Mais além ....

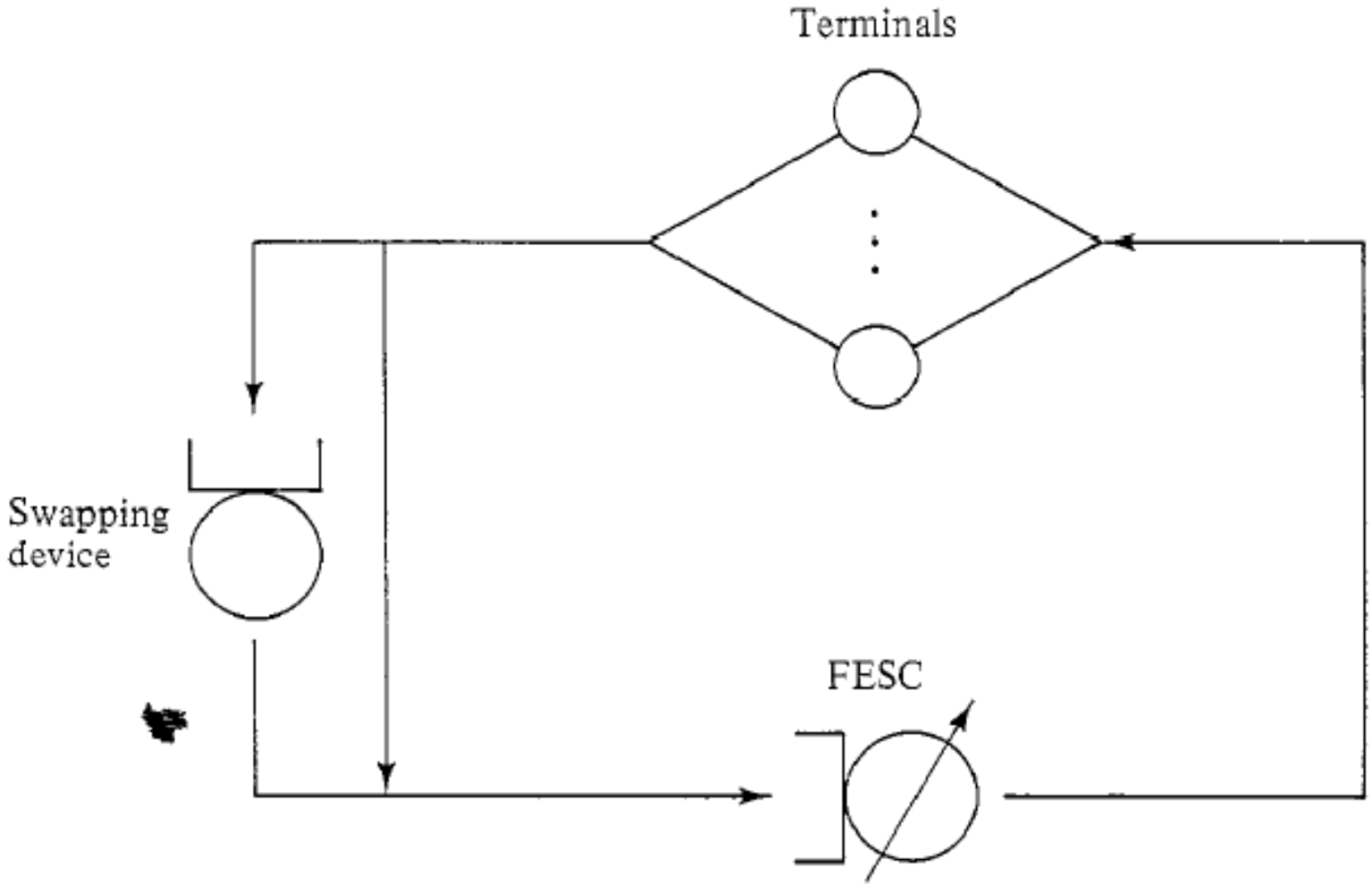


Figure 9.4 – High-Level Model for Swapping to a Dedicated Device

Mais além ....

- Modelos de múltiplas classes com centros LD:
  - generalizações destes modelos com maior complexidade

    (vide livros do Lazowska e Virgílio)
- Modelos não separáveis (p.ex: priority scheduling):
  - Modelos específicos + modificações do algoritmo de MVA apresentado para refletir aspectos não separáveis : soluções aproximadas para modelo exato

    ou
  - Modelos de rede de filas não separáveis + técnicas alternativas que provêm soluções exatas: alto custo computacional
    - Técnica analítica geral para avaliar redes fechadas não separáveis: equilíbrio global (vide Lasowska 8.5)

Exercícios

- Considere um servidor de banco de dados com 3 processadores e 2 subsistemas de disco. Os 3 processadores permitem que o servidor execute processos concorrentes. O uso de um benchmark OLTP indicou a seguinte função como uma boa aproximação do fator de speedup sobre a configuração com um único processador

$$\alpha_{processor}(j) = \begin{cases} 0.56 + 0.44\ j & j \leq 3 \\ 1.88 & j > 3 \end{cases}$$

Sabendo que as demandas por serviço de uma transação típica nos dois subsistemas de disco são 0.008 e 0.011 segundos, respectivamente, e que a demanda por processamento é de 0.033 segundos, qual o impacto de aumentar o número de processos simultâneos no servidor de 3 para 5? Voce consideraria aumentar o número de processadores? Por quê?

Solução

j	alpha(j)				
1	1				
2	1,44				
3	1,88				
4	1,88				
5	1,88				
6	1,88				
7	1,88				
8	1,88				
			População n		
	1	2	3	4	5
Rd1	0,008	0,00923077	0,01031849	0,01140376	0,0123089
Rd2	0,011	0,01332692	0,01560257	0,01807687	0,02039171
Rcpu	0,033	0,04114423	0,0468349	0,05520917	0,06556457
R	0,052	0,06370192	0,07275596	0,08468981	0,09826519
X	19,2307692	31,3962264	41,2337373	47,2311867	50,8827181
Qd1	0,15384615	0,28981132	0,42546993	0,53861309	0,62631053
Qd2	0,21153846	0,41841509	0,64335211	0,85379217	1,03758583
Qcpu	0,63461538	1,29177358	1,93117796	2,60759474	3,33610364
p(1|1)	0,63461538	p(0|1)	0,36538462		
p(1|2)	0,37856604	p(2|2)	0,45660377	p(0|2)	0,16483019
p(1|3)	0,22428663	p(2|3)	0,35772212		
p(3|3)	0,33048236	p(0|3)	0,08750888		
p(1|4)	0,1363939	p(2|4)	0,24276367	p(3|4)	0,29657241
p(4|4)	0,27398907	p(0|4)	0,05028096		
p(1|5)	0,08442824	p(2|5)	0,15904378	p(3|5)	0,21682537
p(4|5)	0,26488486	p(5|5)	0,24471446	p(0|5)	0,03010329

Exercícios

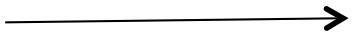
- Considere um servidor com uma CPU e dois discos. Uma requisição requer em média 4 segundos em um disco, 5 segundos no outro disco e 6 segundos na CPU. Suponha que este servidor atenda uma população de 5 máquinas clientes, cada uma com think time médio de 10 segundos. Entretanto, a memória disponível no servidor é tal que no máximo 3 requisições podem executar simultaneamente. A quantidade de memória é suficiente para garantir um tempo de resposta médio inferior a 30 segundos? Qual a chance de haver contenção e espera por memória???

  Você consegue garantir um tempo de resposta médio inferior a 30 segundos se dobrar a memória?

Solução

MVA com 3 centros e N = 3: obtenha X(n)

Dd1 =	4		
Dd2	5		
Dcpu	6		
		População n	
	1	2	3
Rd1	4	5,066667	6,01324503
Rd2	5	6,666667	8,31125828
Rcpu	6	8,4	11,0066225
R	15	20,13333	25,3311258
X	0,06666667	0,099338	0,11843137
Qd1	0,26666667	0,503311	0,71215686
Qd2	0,33333333	0,662252	0,98431373
Qcpu	0,4	0,834437	1,30352941



n	X(n)	alpha(n)	
1	0,0666667		1
2	0,0993377	1,490066225	
3	0,1184313	1,776470588	

$$D = \frac{1}{\mu(1)} = \frac{1}{X(1)} = \frac{1}{0.00667} = 15$$

Resolvendo MVA com 1 centro LD, um delay center (Z=10) e n=5, temos:

R = 33.2659 X = 5/(33.2659+10) = 0.1156

P(1|5) = 0.021296 P(2|5) = 0.0857 P(3|5) = 0.2168
P(4|5) = 0.3654 P(5|5) = 0.3079 P(0|5) = 0.002839

R < 30 segundos??? Não
Probabilidade de espera = P(4|5) + P(5|5) = 0.67

Solução

Se dobrar memória: nunca há contenção por memória
(assumindo N = 5)

MVA com 3 centros LI, N = 5 e Z = 10

População n

	1	2	3	4	5
Rd1	4	4,64	5,32193732	6,0185502	6,703093
Rd2	5	6	7,13675214	8,38361099	9,706632
Rcpu	6	7,44	9,17948718	11,2225124	13,5605
R	15	18,08	21,6381766	25,6246736	29,97022
X	0,04	0,071225	0,09482215	0,11228173	0,125093
Qd1	0,16	0,330484	0,50463755	0,67577323	0,838511
Qd2	0,2	0,42735	0,6767222	0,94132635	1,214233
Qcpu	0,24	0,529915	0,87041873	1,26008311	1,696325

R < 30 segundos??? Sim