Sistemas Operacionais
Quarta Lista de Exercícios – Solução

Norton Trevisan Roman

25 de setembro de 2012

1. $\frac{512-128}{128} = 3$

2. Nesse caso, $p = 0,8$, e $n = 3$

   (a) $1 - 0,8^3 = 1 - 0,512 = 0,488 = 48,8\%$

   (b) Com 1024MB de RAM, poderíamos ter $\frac{1024-128}{128} = 7$ processos na memória. Isso levaria a um uso de $1 - 0,8^7 \approx 1 - 0,21 = 0,79 = 79\%$

   (c) Com $1024+512 = 1536$MB de RAM, poderíamos ter $\frac{1536-128}{128} = 11$ processos na memória. Isso levaria a um uso de $1 - 0,8^{11} \approx 1 - 0,086 = 0,914 = 91,4\%$

   (d) Com a primeira adição de 512MB, gastamos R$80,00 e tivemos um aumento de processamento de $\frac{79}{48,8} \approx 1,619$, ou seja, um ganho de 61,89%. Com a segunda adição, gastamos os mesmos R$80,00 para um aumento de $\frac{91,4}{79} \approx 1,157$, ou seja, um ganho de somente 15,7% – aproximadamente um quarto ($\frac{15,7}{61,89} \approx 0,25$) do ganho anterior. Esse aumento bem mais modesto pode não valer o investimento.

3. Supondo a distribuição aleatória de lacunas e segmentos de dados, podemos assumir que teremos que compactar todo o conteúdo (especialmente se a palavra 0 pertencer a uma lacuna, e a mais alta contiver dados). Isso envolve, para cada palavra de memória (com $\frac{32}{8} = 4$B), dois acessos (um para leitura e um para escrita em nova posição). Como cada operação demora 10ns, cada operação de transferência levará 20ns. Como há $\frac{128\times1024\times1024}{4} = \frac{134.217.728}{4} = 33.554.432$ palavras, temos um gasto de $33.554.432 \times 20 = 671.088.640ns \approx 671ms$. Naturalmente, os bytes pertencentes ao buraco no início da memória não serão movidos (mas entraram na conta acima). Contudo, como há muitas lacunas, podemos supor que as lacunas são pequenas, e o cálculo não será afetado de maneira significativa.

4. Na lista ligada, cada segmento corresponderá a um nó. Como temos $\frac{128MB}{64KB} = \frac{131.072KB}{64kB} = 2.048$ segmentos, a lista terá 2.048 nós. Com 32b = 4B cada nó, teremos $4 \times 2.048 = 8.192 = 8kB$ de espaço ocupados pela lista.

   O mapa de bits, por outro lado, não depende do tamanho de cada segmento (ou de seu número), mas sim do tamanho da unidade de memória. Com nB em cada unidade, teremos $\frac{128MB}{nB} = \frac{134.217.728}{n}$ unidades, ou seja, teremos $\frac{134.217.728}{n}$ bits no mapa, ocupando $\frac{134.217.728}{n\times8} = \frac{16.777.216}{n}$ bytes de espaço.

   O ponto de equilíbrio entre ambas as alternativas se dá em $\frac{16.777.216B}{n} = 8.192B \Rightarrow 2.048B = 2kB$. Ou seja, se $n > 2kB$, o bitmap ocupa menos espaço, já se $n < 2kB$, a lista ocupa menos espaço.

5. Considere as lacunas originais (U – o quanto foi usado, L – o que ainda resta livre):

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*

   (a) Teremos

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*
10	0			12	8	9	9								

Ou seja, C, A, D.

(b)

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*
				12	8	10	8			9	0				

Ou seja, C, D, F.

(c)

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*
10	0									9	0	12	0		

Ou seja, G, A, F.

(d)

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*
				12	8	10	8							9	6

Ou seja, C, D, H.

6. (a)

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*
10	0			27	3										

Ou seja, A, C, C'. Note que, quando o segmento C foi usado pela primeira vez, ele criou um novo segmento – C' – de tamanho 10. Este, por sua vez, foi reusado.

(b)

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*
10	0			27	3										

Ou seja, A, C, C'. Nesse caso, após o uso de parte do segmento C, a busca recomeça a partir do novo segmento C'.

(c)

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*
10	0							7	0			10	2		

Ou seja, A, G, E.

(d)

A (10)		*B (4)*		*C (20)*		*D (18)*		*E (7)*		*F (9)*		*G (12)*		*H (15)*	
U	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*	*U*	*L*
				10	10	10	8							7	8

Ou seja, C, D, H.