Um comparativo do algoritmo de substituição de páginas LRU

Vinicius Visconsini Diniz
Ciências da Computação
Universidade Estadual do Oeste do Paraná - UNIOESTE
Cascavel, Brasil
vinicius.diniz1@unioeste.br

Gustavo Alberto Ohse Hanke
Ciências da Computação
Universidade Estadual do Oeste do Paraná - UNIOESTE
Cascavel, Brasil
gustavo.hanke@unioeste.br

Caio Hideki Gomes Shimohiro
Ciências da Computação
Universidade Estadual do Oeste do Paraná - UNIOESTE
Cascavel, Brasil
caio.shimohiro@unioeste.br

Palavras-Chave—paginação, substituição de páginas, páginas, algorítmo LRU, sistemas operacionais.

I. Paginação de memória

A paginação de memória em um sistema operacional é muito importante nos dias atuais, pois ela permite uma diminuição na fragmentação, isto é, espaços não utilizados na memória.

II. Substituição de páginas

A substituição de páginas é necessária, pois a memória de um sistema computacional não é infinita, e ela ocorre quando a memória está sendo utilizada em sua totalidade.

Nestes casos, o que o sistema operacional faz é escolher uma página que não está sendo utilizada para descarregar da memória, e então carrega a página solicitada na mesma.

No entanto, este processo demanda tempo, pois as páginas que não estão alocadas na memória principal estão alocadas em memória secundária, memória essa com muito maior tempo de leitura que à RAM, e é por este motivo que o SO utiliza algoritmos de substituição de páginas.

A. Algoritmos de substituição de páginas

Um algoritmo de substituição de páginas é o código responsável por escolher qual página deverá ser desalocada da memória, é preciso lembrar que softwares em sua grande maioria não são lineares, isto é, podem acessar endereços em uma ordem aleatória, nem sempre crescente.

A) Optimal (OPT)

O algoritmo ótimo desaloca a página que levará mais tempo para ser referenciada novamente.

É um algoritmo impossível, pois não é possível saber a ordem de acessos de páginas até que eles ocorram.

B) Least Recently Used (LRU)

Este algoritmo desaloca a página que não foi acessada por mais tempo, por meio de uma técnica que pode ser flags ou uma estrutura com comportamento fifo, mas que permite mover um elemento para o final da mesma, isto é, o último elemento a ser removido.

C) Outros algoritmos

O nosso comparativo será com o algoritmo LRU, mas vale ressaltar que ele não é o único, como tudo na computação, podemos resolver um problema de várias maneiras, e um dos algoritmos mais simples de substituição de páginas é o FIFO (first in, first out), existe também o LFU (least frequently used), que salva quantos acessos uma página teve, e na hora de remover uma página ele escolhe a com menos acessos.

Cada um destes algoritmos tem vantagens e desvantagens em casos de usos específicos, o FIFO teria a melhor performance em um código perfeitamente sequencial, o LRU também se daria bem em códigos sequenciais e o LFU se daria muito bem em um código com um loop, pois ele removeria as páginas que não são utilizadas dentro do loop.

III. Testando o LRU

Sabemos que para testar um algoritmo de substituição de páginas precisamos comparar o mesmo com um algoritmo perfeito, o que chamamos de algoritmo OPT, como todos os dados que iremos utilizar para testar o algoritmo LRU já estão disponíveis, o que não acontece na vida real, podemos criar o algoritmo de substituição ótimo.

A. Termos necessários

Antes de falarmos sobre que tipos de testes foram feitos, é preciso conhecer alguns termos importantes.

A) Page fault

Page fault é o nome que damos quando uma página solicitada não está presente na memória, ou seja, quando a execução do algoritmo de substituição é disparada.

B) Reference string

A reference string é uma parte dos acessos à memória feitos por um processo qualquer, pode ser de vários tipos:

Sequencial: Acontece quando a ref string tem a forma x, x+c, x+c+c2, x+c+c3...

Note que nesse caso os endereços das páginas sempre estarão em ordem crescente.

Aleatória: a mais comum, os endereços estão em uma ordem “aleatória”, não parece existir padrão entre os endereços.

Aleatória repetitiva: os endereços são acessados de forma “aleatória”, mas em algum momento eles retornam ao primeiro endereço da sequência e ela se repete por um número indeterminado de vezes.

Estes serão os tipos mais comuns de ref strings.

A nossa ref string não repete valores de páginas, isto é, se uma mesma página for acessada várias vezes em sequência (muito comum), ela aparecerá uma vez na ref string, e só será adicionada novamente na string caso seus acessos estejam

intercalados com acessos a outras páginas, afinal, esse tipo de acesso nunca geraria um page fault.

C) Frames

Frames são regiões de memória do mesmo tamanho de uma página, e são em frames que as páginas são alocadas.

B. Dados Utilizados

Os dados utilizados foram os arquivos trace fornecidos pelo professor, não sei de onde estes arquivos foram obtidos.

O nosso gerador de ref string espera um número referente ao arquivo trace à utilizar, e um valor em porcentagem para copiar para a ref string, no nosso caso, utilizamos o intervalo entre 10% e 100% de cada log, com incrementos de 10%.

C. Os testes

Já sabemos tudo que precisamos, como os dados foram processados, e todos os termos necessários.

Então podemos finalmente comparar o algoritmo LRU com o algoritmo OPT, a saída do nosso sistema foi a quantidade de page faults que ocorreram em uma determinada ref string de um log específico.

Já a nossa entrada foi a indicação da ref string à ser utilizada, a quantidade de frames livres, isto é, quantos frames disponíveis temos na memória no início da execução do processo, e qual algoritmo utilizar, LRU ou OPT.

Com um parser simples podemos transformar o conteúdo de 320 arquivos em uma planilha de Excel. A planilha com todos os resultados se encontra no documento em anexo na pasta do trabalho.

Podemos notar algumas coisas, o OPT tem aproximadamente 94% de page faults para todas as ref strings, já o LRU está muito próximo a 100% de faults, quanto maior a quantidade de frames, menor a quantidade de faults que ocorreram.

IV. Conclusões

Todas estas características parecem indicar algumas coisas:

[1] Mesmo o algoritmo ótimo tem uma quantidade significativa de faults, o que pode indicar uma quantidade grande de páginas sendo referenciadas.

[2] O LRU tem uma quantidade muito grande de faults, demonstrando que ele é pior que o OPT (esperado), mas também mostrando que nestes logs não parece haver um loop significativo, ou pelo menos não existe nem um loop que utiliza menos do que 32 páginas, o máximo de frames livres testados.

[3] Ambos os algoritmos melhoraram quando a quantidade de frames foi aumentada, o que parece indicar que eles não sofrem da anomalia de Belady, e este fato é comprovado verificando as características que causam tal anomalia, e percebemos que tanto o OPT, tanto o LRU não possuem estas características.

Este número reduzido de conclusões é causado por uma quantidade pequena de logs, e também uma quantidade bastante reduzida de frames em memória, sendo 32 frames um valor bastante pequeno de memória livre (esperado, pois estes algoritmos só serão executados caso a memória esteja completamente cheia.), no entanto, atingimos o nosso objetivo principal, que era descobrir como tais algoritmos performariam nos logs fornecidos.

E considerando que mesmo o algoritmo ótimo performou de maneira bastante pobre para as ref strings utilizadas, podemos dizer que o LRU não seria um algoritmo ruim para a substituição de páginas.