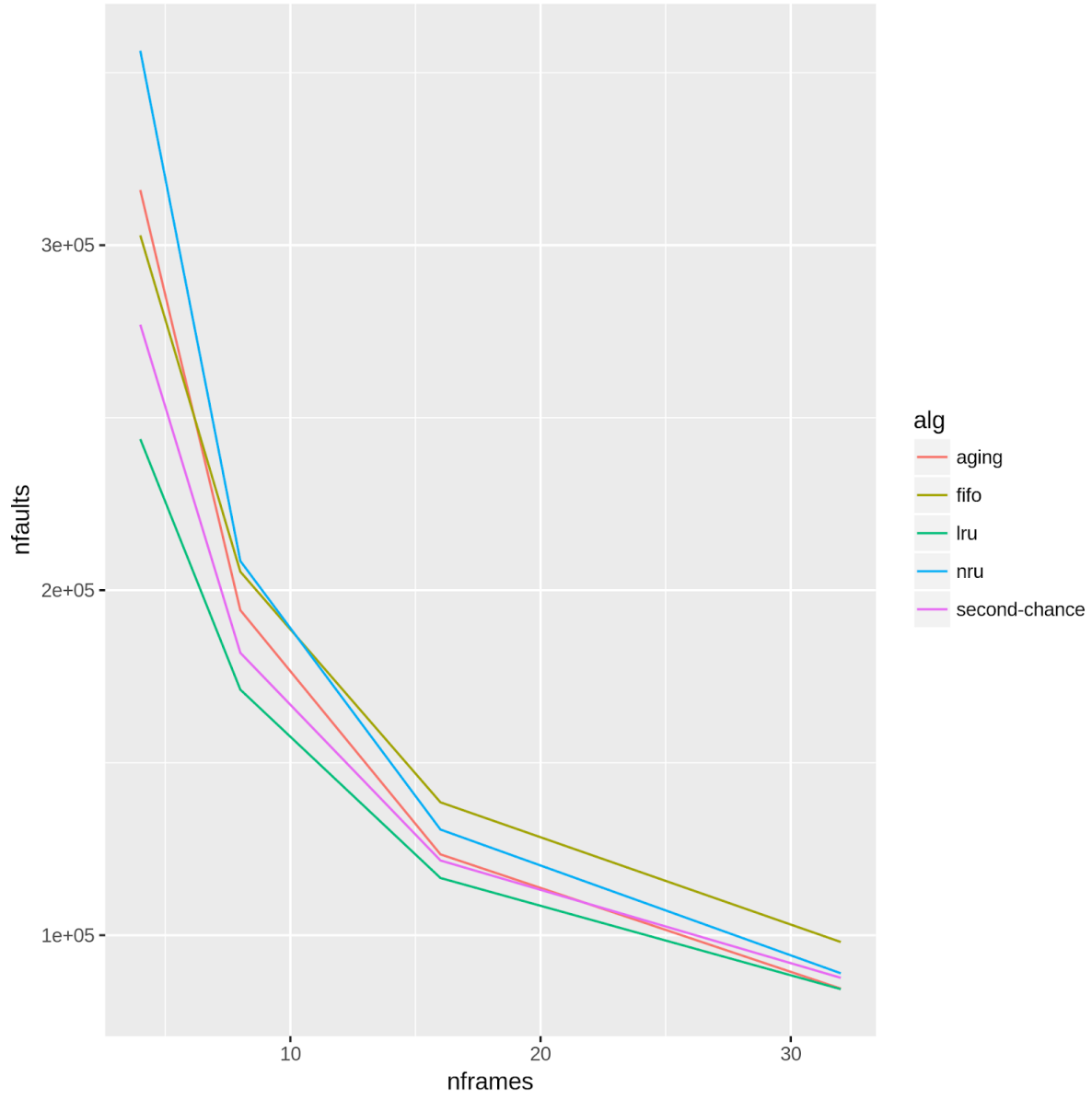
Laboratórios de Projeto de Sistemas Operacionais

Relatório do Lab 4

Simulações

Trace 1

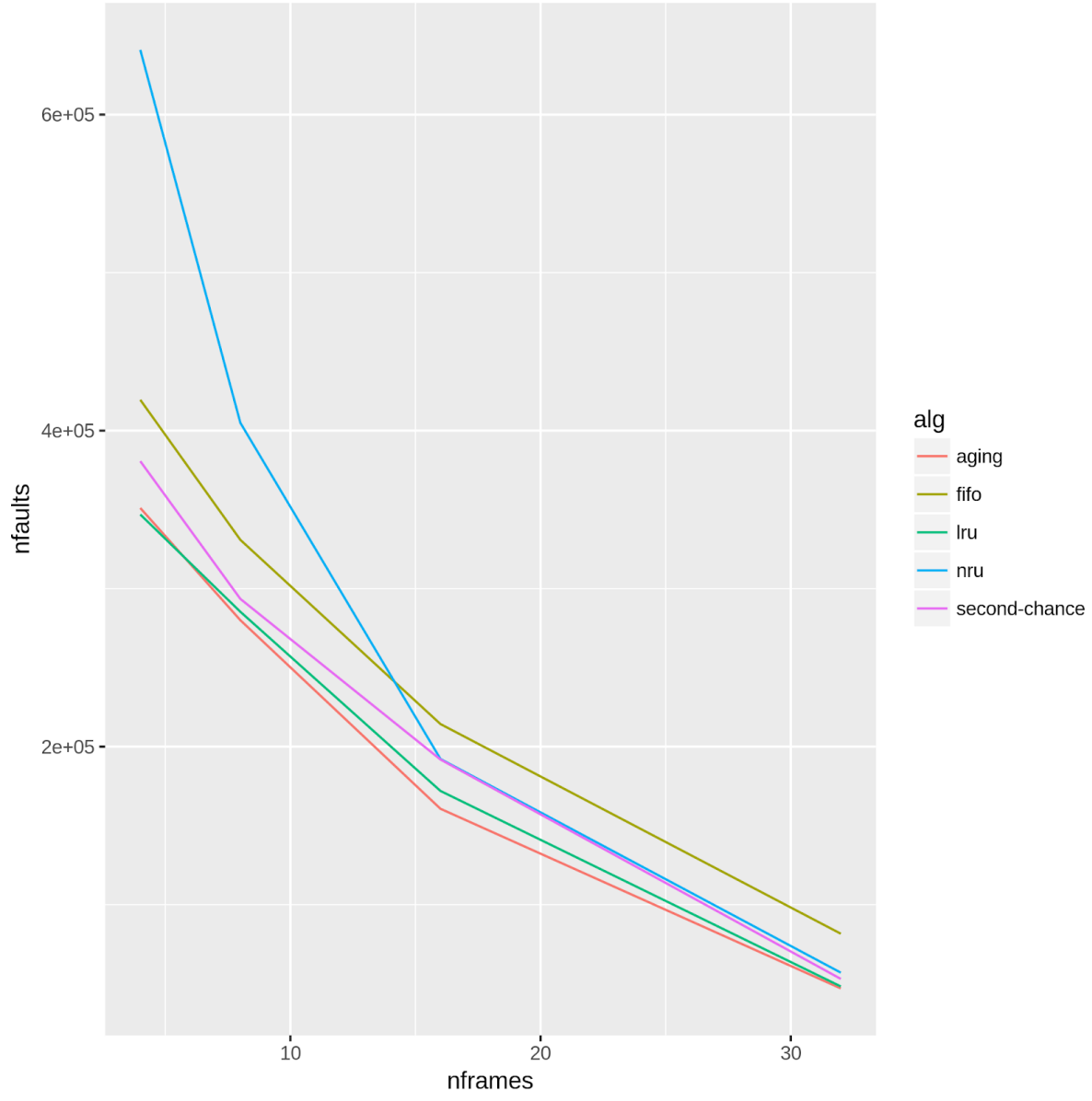


Observado nos plots

- **LRU** é o melhor algoritmo para todos os números de páginas utilizados (de 4 a 32).

- **Second Chance** é o segundo melhor até 25 páginas e após este valor, fica como terceiro melhor até o número máximo de páginas simulado.
- **Aging** é o terceiro melhor entre 5 e 25 páginas, aproximadamente, e segundo melhor para 25 em diante. É possível observar que a curva do **Aging** apresentará menor page fault que o **NRU** caso mais páginas sejam utilizadas.
- **NRU** é o pior até 10 páginas, onde então fica na 4º posição até o número máximo de páginas simulado.
- **FIFO** é o pior dos algoritmos simulados, apesar de até 10 páginas manter-se na 4º posição.

Trace 2



Observado nos plots

- **Aging** é o melhor para todos os número de páginas utilizados (de 4 a 32).
- **LRU** é o segundo melhor para todos os número de páginas utilizados. É possível observar que a curva do **LRU** apresentará menor page fault que o **Aging** caso mais páginas sejam utilizadas.
- **Second Chance** é o terceiro para todos os número de páginas utilizados.
- **NRU** é o pior até aproximadamente 15 páginas, onde então fica na 4º posição até o número máximo de páginas simulado. **NRU** pouco depois de 15 e antes de 25 fica bem próximo do **Second Chance**.
- **FIFO** é o pior dos algoritmos simulados, apesar de até pouco depois de 15 páginas manter-se na 4º posição.

Explicação

Em ambos os experimentos o algoritmo FIFO teve um desempenho tão mais ou tão quanto inferior aos demais. Tal fato pode ser explicado por sua extração bruta de páginas, ou seja, mesmo páginas que são frequentemente utilizadas podem ser retiradas da memória principal pelo fato de terem chegado primeiro na fila ou páginas mais novas não serem referenciadas nas próximas instruções mas se manterem persistentes pois o procedimento clássico só leva em consideração o tempo que uma página persiste em memória principal (o início da fila contém a página mais antiga e o final a mais nova).

Entretanto, apesar do algoritmo NRU ser mais inteligente e adequado, sua implementação clássica depende de um controle de bits em hardware, o que causa um uso de memória para representar tais elementos quando em simulação. De fato, este procedimento se colocou como tolerável em desempenho mas pode ter tido seu desempenho menor em comparação aos outros pela quantidade de operações necessárias para efetuar seu trabalho, como por exemplo a atualização dos bits de controle a cada referência feita a memória.

Algumas alterações na estrutura natural do FIFO são feitas para torná-lo útil. O Second Chance é um exemplo de algoritmo inspirado no FIFO. Nos experimentos foi possível notar um desempenho mais adequado em desempenho que o NRU, principalmente para menos quadros de memória. Este algoritmo utiliza um bit de flag para sinalizar se a página mais antiga é pouco utilizada ou não, mecanismo que reduz o *overhead* existente no FIFO puro.

O desempenho do LRU e Aging em ambos os experimentos pode ser explicado por sua aproximação com a natureza da estratégia ótima: levam em consideração a frequência de utilização de uma página, dessa forma tendo uma intuição sobre a decisão de retirá-la ou não, num dado momento, da memória. Pois *páginas que foram utilizadas intensamente nas últimas instruções provavelmente o serão em seguida de novo.*

Comparação na literatura

A perspectiva de um algoritmo ótimo de substituição de páginas pode nortear a decisão acerca de qual algoritmo útil em sistemas reais é reconhecido como mais apropriado, por aproximação, dentre os mais comuns na literatura, alguns aqui referenciados nos experimentos. Um procedimento ótimo para troca de páginas deveria ser capaz de prever quando cada página será referenciada em seguida e assim retirar da memória a página que mais tempo ficaria ociosa na memória. No entanto, quando ocorre falta de página o sistema operacional não tem como saber quando cada página será referenciada.

Portanto, segue uma comparação com alguns prós e contras de cada algoritmo mais comum retirada do livro Sistemas Operacionais Modernos do Tanebaum.

Algoritmo	Comentário
Ótimo	Não implementável, mas útil como um padrão de desempenho
NRU (não usado recentemente)	Aproximação muito rudimentar do LRU
FIFO (primeiro a entrar, primeiro a sair)	Pode descartar páginas importantes
Segunda chance	Algoritmo FIFO bastante melhorado
Relógio	Realista
LRU (usada menos recentemente)	Excelente algoritmo, porém difícil de ser implementado de maneira exata
NFU (não frequentemente usado)	Aproximação bastante rudimentar do LRU
Envelhecimento (*aging*)	Algoritmo eficiente que aproxima bem o LRU
Conjunto de trabalho	Implementação um tanto cara
WSClock	Algoritmo bom e eficiente

Algoritmos

FIFO

Descrição

O SO mantém uma fila das páginas correntes na memória. A página no início é a mais antiga e a página no final é a mais recente. Quando ocorre um page fault a página do início é removida e a nova é inserida no final da fila.

Implementação

Put: Adiciona a página nova ao final da fila

Evict: Remove a página mais antiga e retorna seu id
Clock e **Access** não são utilizados

Second Chance

Descrição

O algoritmo second chance é uma FIFO melhorada, isso porque ela segue a mesma lógica de verificar o início da fila, além de verificar o bit R. Se o bit R estiver em 0 a página é removida e o seu Id é retornado, se o bit estiver setado em 1, ele passa a ser 0 e é adicionado no final da fila como se fosse uma nova página e a execução segue procurando a primeira página com bit R = 0. Se todas as páginas estiverem com o seu bit R em 0, o second chance funcionará como um FIFO.

Implementação

Put: Adiciona a página nova ao final da fila com bit de uso 0.
Evict: Remove a página mais antiga que o bit R é 0.
Access: Quando uma página é utilizada o bit R dela é setado para 1.
Clock: Não é utilizado.

LRU

Descrição

No *Least Recently Used* a idéia é que as páginas que foram utilizadas nas últimas instruções provavelmente serão usadas novamente nas próximas. O LRU utiliza um timer que é atualizado a cada que vez que uma página é acessada, logo, a página com o menor timer será a que foi menos utilizada visto que esse *timer* não foi atualizado para um valor mais recente.

Implementação

Put: Temos a variável timer, que é uma variável global, o put incrementa esse valor do timer e adiciona a página a tabela.
Evict: Procura a página com o menor timer, remove e retorna seu índice.
Access: Sempre que uma página é acessada, atualiza o valor timer dela.
Clock: Não é utilizado.

NRU

Descrição

Not Recently Used utiliza dois bits, os bits R e M que indicam quando uma página foi lida ou escrita (R) ou quando ela foi apenas escrita (M). Quando uma página é criada os bits R e M são 0 e o bit R é setado para 0 a cada interrupção de relógio, o bit M é mantido, para que o SO saiba se deve escrever a página no disco. Quando ocorre um page fault remove uma página aleatoriamente das classes mais baixas(bits 00, 01, 10, 11).

Implementação

Put: Adiciona a página com os bits R e M em 0.
Evict: Criamos arrays para cada classe a adicionamos os valores nesses arrays, depois verificamos se na classe mais baixa tem páginas disponíveis para remoção, se sim, remove aleatoriamente uma página e retorna seu Id, caso não tenha, verifica na próxima classe até uma página ser encontrada e removida.
Clock: Sempre que o clock é invocado fazemos um for, varrendo a tabela de frames e setamos o bit R de todas as páginas para 0.
Access: Sempre que uma página é acessada, setamos o bit R para 1 e verificamos se essa página está sendo escrita, se sim, setamos o bit M para 1.

Aging

Descrição

Cada frame terá um contador, inicialmente com valor um. A cada interrupção do clock, o SO varre as páginas da memória deslocando os contadores a direita em um bit. Quando ocorre um page fault, a página com menor contador é removida. Quando uma página é acessada, o contador é atualizado adicionando 1 ao bit mais à esquerda.

Implementação

Put: Adicionamos a página com o contador iniciado em 1.
Evict: Percorremos a tabela procurando a página com o menor valor para o contador, removemos ela da tabela e retornamos seu Id.
Clock: Fazemos um shift à direita do contador da página, para todas as páginas.
Acess: Procuramos a páginas quando a encontramos utilizamos o operador binário ou entre o contador e do bit mais à esquerda.