

Gerenciamento de memória virtual

1.1 Enunciado

O desenvolvedor de um sistema operacional deseja avaliar o impacto (a) do algoritmo de substituição de páginas, (b) do tamanho da página e (c) da velocidade do disco sobre o desempenho do sistema de gerenciamento de memória virtual por paginação em um nível, que é medido como o tempo médio de acesso a um dado referenciado na memória. O sistema é composto pelas referências a páginas virtuais (que contém os dados referenciados), que são parte dos endereços virtuais gerados pelos processos em execução. O mecanismo de hardware de memória virtual deve verificar se a referência está na TLB (uma cache do mecanismo de paginação), cujo acesso leva 1ns e, se estiver, há um acerto na TLB e a página referenciada está na memória RAM. Neste caso, basta um único acesso à memória RAM (cujo acesso leva 10ns) e o dado é acessado. Se não houver acerto na TLB, o hardware deve acessar a tabela de páginas daquele processo na memória RAM. Se a tabela de páginas indicar que a página referenciada é válida, então ela está na memória RAM. Neste caso, é necessário atualizar a TLB (1ns) e fazer um novo acesso à memória RAM para acessar o dado. Porém, se a tabela de páginas do processo informar que a página não é válida, então ela deve ser buscada do disco. Neste caso, o processo deve ser bloqueado, ou seja, impedido de continuar executando, e a página faltante do disco deve ser transferida para a memória principal. O tempo de transferência é de 1us por KB transferido para discos lentos e de 500ns/KB para discos rápidos (níveis mínimo e máximo). Se houver páginas físicas (frames) livres na memória principal, então o sistema operacional escolhe uma frame livre qualquer, transfere a página do disco para aquela frame e, ao término da transferência, atualiza a tabela de páginas do processo, conclui o acesso ao dado faltante (um acesso à RAM) e desbloqueia o processo. Se a memória física tiver todos seus frames alocados, então uma página física (frame) deve ser escolhida para ser removido da memória (frame vítima), conforme um algoritmo de substituição de páginas. A política de substituição de páginas é local. Se a página física tiver sido alterada, ela primeiro deve ser transferida inteiramente para o disco antes de receber a nova página virtual. Os algoritmos de substituição de páginas a serem avaliados são o LRU (Least Recently Used), que substitui a página referenciada a mais tempo no passado e o LFU (Least Frequently Used), que substitui a página menos frequentemente usada. A memória física possui 256KB. As páginas podem variar de 4KB a 16KB (níveis mínimo e máximo). Há 2 processos executando. Cada processo em execução (não bloqueado) gera uma referência à memória apenas após sua última referência ter sido atendida, após um atraso que foi medido muitas vezes (em ns) e que está no arquivo "tempo_entre_referencias_memoria_ns.txt". 98% das referências a páginas virtuais são para a última página virtual referenciada pelo processo, e o restante para uma página virtual aleatória qualquer. A quantidade de páginas virtuais depende do tamanho da página, sendo que os endereços virtuais possuem 16 bits. Além de avaliar o impacto dos fatores citados sobre a métrica de desempenho, o desenvolvedor do sistema operacional deseja obter estatísticas sobre o tempo em que os processos ficaram bloqueados, o tempo médio de

transferência de uma página e ainda sobre a taxa de ocupação da memória física. Ele também gostaria de saber se pode afirmar, com 90% de certeza, que o tempo médio que processos ficam esperando (bloqueados) pelo gerenciador de memória é diferente para cada algoritmo avaliado, e se páginas de 4KB levam a menos tempo de bloqueio dos processos que páginas de 16KB. Simule esse sistema por 1 hora, com tempo de aquecimento de 10%.

1.2 Relação dos objetivos (requisitos funcionais e não funcionais).

Os seguintes objetivos já foram modelados no arena e já se tem a resposta, que é avaliar o impacto de:

- Algoritmo substituição de páginas
 - Tamanho das páginas
 - Velocidade de disco sobre o desempenho do sistema de gerenciamento memória virtual em um nível.
- (tempo médio de acesso a um dado referenciado na memória).

Então agora a partir desses objetivos, que já estão modelados no arena, resta desenvolver os componentes no genesys, e modelar no genesys e verificar corretude dos resultados.

RFs

Implementar e testar componentes que são do modelo de acesso a memória.

- Testar componente já existente “Station”
- Implementar componente de “Seize”.
- Implementar componente “Delay”.
- Testar componente já existente “Search”.
- Implementar componente “Assign”.
- Implementar componente “Decide”.
- Implementar componente “Route”

Implementar e testar componentes que são do modelo de Loop de um processo

- Implementar componente “Hold”
- Testar componente “Separate”

Implementar bloco de “while” que é utilizado nos modelos dos algoritmos

Implementar componentes “Init” e “Dispose” que são dos modelos de criação de processos e fim de acesso

RNFs

Modelar no genesys modelos análogos aos que já existem no arena.

Realizar simulação do modelo que foi simulado no arena, no genesys.

Realizar testes de hipótese para validar os resultados das simulações, e comprovar corretude das implementações.

1.3 Execução.

1ª Etapa: Realizar simulação no arena.

Primeiramente será realizada a simulação no arena para coletar os dados e resultados da simulação. Simplesmente executando o modelo já existente no arena, esses dados serão utilizados posteriormente para validar os componentes criados.

2ª Etapa: Implementar componentes *seize* e *delay*

Nessa etapa serão implementados o componente *seize*, e o componente *delay*. Esses componentes são necessários para a implementação do modelo de acesso a memória.

3ª Etapa: Implementar componentes *assign* e *decide*

Implementação de componentes de *assign* e *delay*, que são mais componentes que são necessários no modelo de acesso a memória.

4ª Etapa: Implementar componente *route*

Implementação do último componente que falta para o modelo de acesso a memória.

5ª Etapa: Testar componentes *search* e *station*

Após a implementação dos últimos componentes, esses dois componentes já existentes podem ser validados por meio do modelo de acesso a memória, e corrigidos caso seja necessário.

6ª Etapa: Implementar componentes *init* e *dispose*

Esses componentes são necessários para os modelos de fim de acesso a memória, e de criação de processos, que são pequenos porém cruciais para o funcionamento do modelo do sistema como um todo.

7ª Etapa: Implementar componente *hold*

Nessa etapa, se dá início a implementação do componente *hold* que é parte do modelo de loop do processo. Combinado com os componentes já feitos nas etapas anteriores este é o único componente novo.

8ª Etapa: Testar componente *separate*

Dado os componentes desenvolvidos anteriormente, e o componente feito na última etapa, o modelo de loop do processo. O componente *separate* já está implementado, e por meio do modelo de loop do processo ele será validado e corrigido caso necessário.

9ª Etapa: Implementar bloco de *while*

Para o modelo do algoritmo de substituição de página só faltam os blocos *while* e *endwhile*, que serão implementados e possibilitam o modelo ser importado para o genesys.

1.4 Testes.

Após a etapa 6, os modelo de acesso a memória, criação de processos já poderam ser implantados no genesys de maneira análoga ao que existe no arena. Estes serão executados, e tendo os resultados do arena da execução destes, será feita uma comparação dos dados obtidos..

Após as etapa 8 e 9 o modelos de loop dos processos e algoritmos de substituição de páginas poderão ser implantados no genesys, dessa forma o modelo completo do sistema vai estar no genesys, e este será executado e os resultados obtidos comparados com o que se tem no arena.

1.5 Estimativa do tempo

ATIVIDADE	TEMPO
Etapa 1	2 horas
Etapa 2	4 ~ 5 horas
Etapa 3	4 ~ 5 horas
Etapa 4	2 ~ 3 horas
Etapa 5	2 ~ 3 horas
Etapa 6	4 ~ 5 horas
Teste 1	3 horas
Etapa 7	2 ~ 3 horas
Etapa 8	2 ~ 3 horas
Etapa 9	2 ~ 3 horas
Teste 2	4 horas

1.6 Cronograma

ATIVIDADES	AGO	SET	OUT	NOV
------------	-----	-----	-----	-----

Primeira entrega parcial	26/08			
Desenvolvimento da etapa 1	x	x		
Desenvolvimento da etapa 2	x	x		
Desenvolvimento da etapa 3		x		
Segunda entrega parcial		09/09		
Desenvolvimento da etapa 4		x		
Desenvolvimento da etapa 5		x		
Desenvolvimento da etapa 6		x		
Terceira entrega parcial		23/09		
Realização do teste 1		x	x	
Desenvolvimento da etapa 7		x	x	
Desenvolvimento da etapa 8			x	
Quarta entrega parcial			07/10	
Desenvolvimento da etapa 9			x	
Realização do teste 2			x	
Pendências e correções de código			x	
Quinta entrega parcial			21/10	
Correções de código, correções no relatório, etc.				
Entrega final				04/11
Ajustes entrega final				16/11

1.7 GitHub

gstvob / **ModSim**

Unwatch 1

Star 0

Fork 0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Security

Insights

Settings

Options

Collaborators

Branches

Webhooks

Notifications

Integrations & services


Deploy keys

Moderation

Interaction limits

Collaborators

Push access to the repository



Prof. Rafael Cancian, Dr. Eng.
Awaiting rlcancian's response

Copy invite link

Cancel invite

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

Add collaborator

gstvob Initial commit, genesys and model

Latest commit 86e6496 2 minutes ago

Genesys

Initial commit, genesys and model

2 minutes ago

Modelo_pronto

Initial commit, genesys and model

2 minutes ago

README.md

Initial commit, genesys and model

2 minutes ago

README.md

ModSim