

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

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

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

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

Implementar componentes “Init” e testar o componente “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.

Para todas as etapas que envolvam tanto um componente a ser criado quanto um componente que já exista, já está previsto no final dela um teste unitário destes componentes.

Esses componentes serão todos testados e implementados antes de serem unificados no modelo final onde provavelmente serão necessários mais testes e mais correções nestes componentes.

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 componente *hold*

Nessa etapa será implementado o componente *hold*. Esse componente será o primeiro a ser implementado, logo será feito em uma etapa sozinho pois ele será o componente de “aprendizado”.

3ª Etapa: Testar os componentes *separate e route*

Realizar testes nos componentes já implementados e corrigir caso necessário. Após essa etapa é possível criar uma

4ª Etapa: Implementar componente *init* e testar componente *dispose*

Esses dois componentes, em conjunto com os que foram implementados e testados nas etapas anteriores, permitem a criação de uma parte do modelo, portanto ao implementá-los será possível modelar essa parte e realizar algum tipo de teste.

5ª Etapa: Testar componentes *search e station*

Outra etapa com componentes que já estão implementados a princípio, portanto nessa etapa serão realizados testes nesses componentes e corrigidos caso necessário.

6ª Etapa: Implementar componentes *while* e testar componente *decide*

Implementação do bloco *while* que é utilizado nas partes dos modelos dos algoritmos.

7ª Etapa: Testar componentes *assign e delay*

Nessa etapa, serão testados os componentes já implementados *assign e delay*, e feitas as correções caso seja necessário.

8ª Etapa: Testar componente *seize*

O último componente a ser testado, o *seize* será testado e corrigido também caso necessário.

Etapa Final: Integrar componentes e corrigi-los se necessários.

Essa etapa será a etapa final, nessa etapa os componentes irão compor o modelo final e este será executado no genesys, caso necessário, nessa etapa serão corrigidos os componentes necessários.

1.4 Testes.

Após todas as etapas que envolvem implementação de componentes, serão realizados testes iniciais nestes componentes com o propósito de dar uma validação inicial aos componentes.

Após a etapa 4, terão sido implementados alguns componentes que permitem a passagem de parte do modelo existente para o genesys, esses modelos parciais irão ser utilizados para realização de alguns testes dos componentes implementados até então, e se necessário, esses componentes serão corrigidos.

Após a etapa 6, uma outra parte do modelo também poderá ser gerada no genesys, e como na etapa 4, esta parte será utilizada com o propósito de validar os componentes que foram implementados até o momento.

Na última etapa, o modelo será então juntado como um todo, e o teste final será realizado, corrigindo componentes caso necessário.

1.5 Estimativa do tempo

ATIVIDADE	TEMPO
Etapa 1	2 horas
Etapa 2	5 horas
Etapa 3	4 horas
Etapa 4	4 horas
Teste de integração 1	5 horas
Etapa 5	3 horas
Etapa 6	3 horas
Teste de integração 2	5 horas
Etapa 7	3 horas
Etapa 8	3 horas
Final	15 horas

1.6 Cronograma

Semana	Etapas
Semana 1 - Primeira entrega parcial (26/08)	X
Semana 2	E1 e E2
Semana 3 - Segunda entrega parcial (11/09)	E1 e E2
Semana 4	E3 - E4
Semana 5 - Terceira entrega parcial (23/09)	E4 - E5
Semana 6	E6 - E7 E8
Semana 7 - Quarta entrega parcial (07/10)	E6 - E7 E8
Semana 8	EFinal
Semana 9 - Quinta entrega parcial (21/10)	EFinal
Semana 10	Correções
Semana 11 - Entrega final (04/11)	Correções

1.7 GitHub


9 commits

1 branch









0 releases



1 contributor

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find File](#) [Clone or download](#)

 **gstvob** Adicionando componentes alterados, e entregaveis da entrega passada

Latest commit 16c5031 2 minutes ago

 Componentes_implementados	Adicionando componentes alterados, e entregaveis da entrega passada	2 minutes ago
 Genesys	Adicionando signal e hold	12 hours ago
 Modelo_pronto	Initial commit, genesys and model	last month
 .gitignore	Adicionando signal e hold	12 hours ago
 README.md	Initial commit, genesys and model	last month
 primeira_entrega_parcial_Gustav...	First partial deliverable .pdf	last month
 segunda_entrega_parcial.pdf	Adicionando componentes alterados, e entregaveis da entrega passada	2 minutes ago
 segunda_entrega_parcial_slides.pdf	Adicionando componentes alterados, e entregaveis da entrega passada	2 minutes ago

 **README.md** 

ModSim