Relatório de Trabalho
Simulação de memória virtual

Membros: Luigi Perotti Souza - 201910462
Professor: Benhur Stein

17 de março de 2021

Introdução

O seguinte relatório tem por finalidade descrever o desenvolvimento e resultados referentes ao segundo Trabalho da Disciplina de Sistemas Operacionais. Para o desenvolvimento do programa foi utilizada a ferramenta Intellij IDEA, onde todo o código foi escrito na linguagem java com auxílio da ferramenta Json Simple disponível em: https://code.google.com/archive/p/json-simple/ para codificação e decodificação de arquivos .json, utilizados na leitura de Jobs do trabalho.

Metodologia

Os testes realizados, partiram de executar um mesmo conjunto de instruções, porém com diferentes configuralções de execução. Para a execução do programa, todos os processos testados foram previamente carregados em memória, a partir de um arquivo .json com os seguintes parâmetros: 'program', onde será todas a instruções do processo, 'priority', que é a prioridade do processo durante o escalonamento, 'IOSize' a quantia de memória necessária para os campos de entrada e saída e, por fim, um parâmetro 'IO' que se trata de uma lista com os arquivos de E/S contendo um 'cost', que é o tempo de gravação necessária para a leitura ou escrita e a 'data', que é o valor da entrada ou saída. Para os testes foram utilizados os seguintes programas:

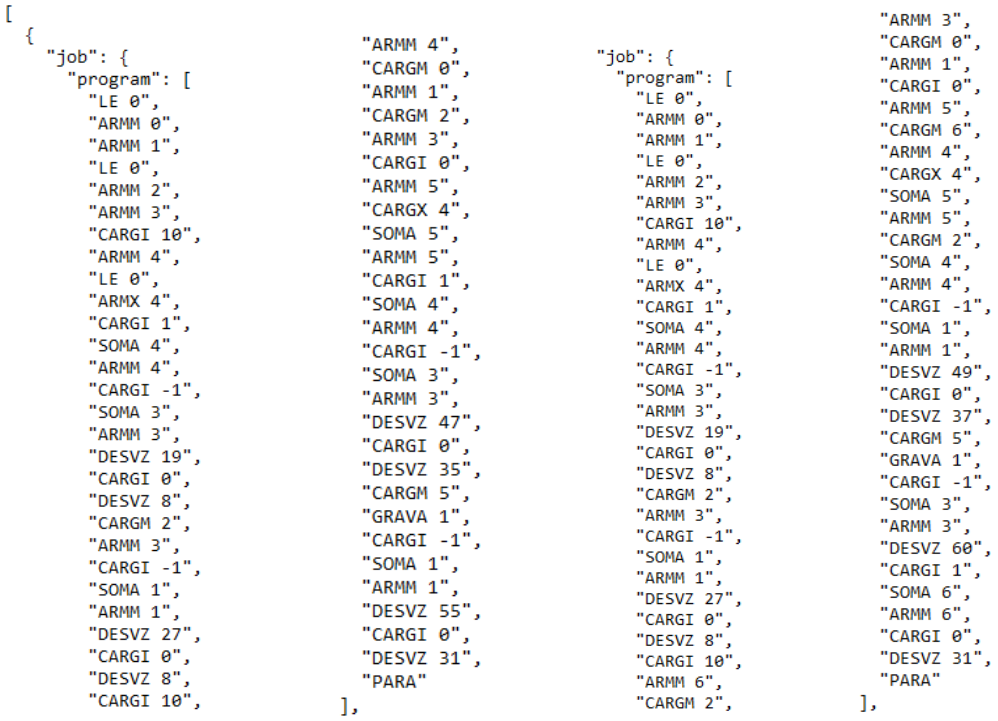

Figura 1: Processos utilizados

Ambos os programas tratam-se de que, a partir de uma matriz de entrada, será feito o somatório de suas linhas ou colunas, sendo o primeiro referente ao somatório das linhas, e o segundo, ao das colunas. Ambos os programas tiveram a mesma matriz de entrada, onde por consequência, o programa 2 necessitou mais tempo para sua execução uma vez que o número de colunas era maior que o de linhas. Por conta disso o programa dois teve uma prioridade de execução maior (0.4) em relação ao primeiro (0.5), onde quanto menor o valor, mais o programa será escolhido para execução. Apesar de tudo, está métrica foi a que menos apresentou relevância, uma vez de se tratar apenas de dois processos, o único caso para tal prioridade apresentar diferença, é caso ambos os processos tenham interrupções ao mesmo tempo. A prioridade manteve-se fixa durante toda a execução e nenhum re cálculo foi feito.

Foram feitos três testes, cada um com uma configuração de memória e configuração de fila de prioridade própria. Na execução dos processos, a paginação se deu a partir da definição de páginas de tamanho 2, ou seja, cada

```
"IO": [
  {
    "cost": "2",
    "data": [
      "3","10",
      "1","2","3","4","5","6","7","8","9","10",
      "1","1","1","1","1","1","1","1","1","10",
      "5","5","4","0","0","2","8","9","2","2"
    ]
  },
  {
    "cost": "3",
    "data": [
      "0"
    ]
  }
]
```


Figura 2: Arquivos de entrada e saída

página na memória armazenará dois números inteiros. Como cada processo necessita de 40 endereços de memória (30 para armazenar a matriz e 10 para os dados de iteração), o tamanho necessário na memória secundária foi de exatamente 20 páginas. Qualquer número diferente desse causará um acesso a uma página não mapeada ou então desperdício de memória.

Para o controle das interrupções, o valor do Quantum foi de 6 instruções, permitindo a rotacionalidade dos processos, e, os custos de escrita da memória principal e de limpeza desta para a secundária, foram respectivamente de 2 e 4 instruções. Como uma chamada de Page Fault que requerer uma limpeza da página da memória principal, eventualmente vai requerer sua escrita logo em seguida, podemos dizer que o custo sempre que uma chamada de Page Fault tiver a posição de memória suja, vai requerer 6 instruções de interrupção.

Para cada teste, foram utilizadas as configurações descritas acima igualmente, diferenciando apenas entre utilizar o método convencional de fila de prioridade (primeiro a entrar, primeiro a sair), e o método da segunda chance. Para cada modelo de fila, foi testado o programa com uma memória principal de capacidade de 40 páginas, suficiente para armazenar ambos processos sem necessidade de liberar qualquer endereço, 20 páginas, onde algumas vezes haverão mudanças e por fim, 3 páginas, onde constantemente haverá rotação de páginas na memória principal. Os processos descritos não suportam uma memória com capacidade de apenas duas ou uma página, uma vez que diversas vezes haverá a referência a endereços não mapeados ou não haverá páginas disponíveis para que os dois processos coexistam.

Para o teste com 3 páginas de memória física, foi utilizado um contador para sabermos quantas vezes um processo tentou validar uma tabela de páginas a principal e falhou tendo em vista as demais estarem com um status 'não alterável'.

Resultados Obtidos

Algoritmo FIFO

Esse exemplo segue as entradas de exemplo na Figura 2, com todas as prioridades descritas anteriormente, utilizando o método padrão de fila de prioridade.

- Para memória de 40 paginas (100%):

Tempo total do Timer:	1694
Tempo Ativo de CPU:	1646
Tempo Ocioso:	48
Chamadas de Sistema:	116
Chamadas do tipo LE:	64
Chamadas do tipo GRAVA:	13
Chamadas do tipo STOP:	2
Chamadas Ilegais	0
Chamada de PageFault:	37
Trocas de Processo:	299
Número de Preempções:	187

Descrição para cada processo (0/1):

Hora de Início:	(0/1)
Hora de Término:	(1545/1694)
Tempo de Retorno:	(1545/1693)
Tempo de CPU:	(764/882)
Tempo Bloqueado por PageFault:	(36/38)
Tempo Bloqueado por Falta de Página:	(0/0)
Tempo Bloqueado por Chamada E/S:	(73/94)
Tempo Total Bloqueado:	(109/132)
Vezes Bloqueado:	(54/62)

Falha de Página:	(18/19)
Falta de Página:	(0/0)
Chamadas E/S:	(35/42)
Tempo de Espera para ser Escalonado:	(672/679)
Vezes Escalonado:	(149/150)
Número de vezes que a CPU foi perdida:	(96/91)

- Para memória de 20 paginas (50%):

Tempo total do Timer:	1817
Tempo Ativo de CPU:	1696
Tempo Ocioso:	121
Chamadas de Sistema:	166
Chamadas do tipo LE:	64
Chamadas do tipo GRAVA:	13
Chamadas do tipo STOP:	2
Chamadas Ilegais	0
Chamada de PageFault:	87
Trocas de Processo:	329
Número de Preempções:	170

Descrição para cada processo (0/1):

Hora de Início:	(0/1)
Hora de Término:	(1716/1817)
Tempo de Retorno:	(1716/1816)
Tempo de CPU:	(788/908)
Tempo Bloqueado por PageFault:	(192/182)
Tempo Bloqueado por Falta de Página:	(0/0)
Tempo Bloqueado por Chamada E/S:	(73/94)
Tempo Total Bloqueado:	(265/276)
Vezes Bloqueado:	(78/88)
Falha de Página:	(42/45)
Falta de Página:	(0/0)
Chamadas E/S:	(35/42)
Tempo de Espera para ser Escalonado:	(663/632)
Vezes Escalonado:	(164/165)

Número de vezes que a CPU foi perdida: (87/83)

- Para memória de 3 paginas (7,5%):

Tempo total do Timer:	2944
Tempo Ativo de CPU:	2112
Tempo Ocioso:	832
Chamadas de Sistema:	582
Chamadas do tipo LE:	64
Chamadas do tipo GRAVA:	13
Chamadas do tipo STOP:	2
Chamadas Ilegais	0
Chamada de PageFault:	503
Trocas de Processo:	503
Número de Preempções:	22

Descrição para cada processo (0/1):

Hora de Início:	(0/1)
Hora de Término:	(2421/2944)
Tempo de Retorno:	(2421/2943)
Tempo de CPU:	(967/1145)
Tempo Bloqueado por PageFault:	(1050/1400)
Tempo Bloqueado por Falta de Página:	(44/8)
Tempo Bloqueado por Chamada E/S:	(73/94)
Tempo Total Bloqueado:	(1167/1502)
Vezes Bloqueado:	(257/325)
Falha de Página:	(221/282)
Falta de Página:	(22/4)
Chamadas E/S:	(35/42)
Tempo de Espera para ser Escalonado:	(287/296)
Vezes Escalonado:	(251/252)
Número de vezes que a CPU foi perdida:	(19/3)

Algoritmo 2ª Chance

Esse exemplo segue as entradas de exemplo na Figura 2, com todas as prioridades descritas anteriormente, utilizando o método de fila com segunda chance.

- Para memória de 40 paginas (100%):

Tempo total do Timer:	1694
Tempo Ativo de CPU:	1646
Tempo Ocioso:	48
Chamadas de Sistema:	116
Chamadas do tipo LE:	64
Chamadas do tipo GRAVA:	13
Chamadas do tipo STOP:	2
Chamadas Ilegais	0
Chamada de PageFault:	37
Trocas de Processo:	299
Número de Preempções:	187

Descrição para cada processo (0/1):

Hora de Início:	(0/1)
Hora de Término:	(1545/1694)
Tempo de Retorno:	(1545/1693)
Tempo de CPU:	(764/882)
Tempo Bloqueado por PageFault:	(36/38)
Tempo Bloqueado por Falta de Página:	(0/0)
Tempo Bloqueado por Chamada E/S:	(73/94)
Tempo Total Bloqueado:	(109/132)
Vezes Bloqueado:	(54/62)
Falha de Página:	(18/19)
Falta de Página:	(0/0)
Chamadas E/S:	(35/42)
Tempo de Espera para ser Escalonado:	(672/679)
Vezes Escalonado:	(149/150)
Número de vezes que a CPU foi perdida:	(96/91)

- Para memória de 20 paginas (50%):

Tempo total do Timer:	1777
Tempo Ativo de CPU:	1678
Tempo Ocioso:	99
Chamadas de Sistema:	148
Chamadas do tipo LE:	64
Chamadas do tipo GRAVA:	13
Chamadas do tipo STOP:	2
Chamadas Ilegais	0
Chamada de PageFault:	69
Trocas de Processo:	323
Número de Preempções:	180

Descrição para cada processo (0/1):

Hora de Início:	(0/1)
Hora de Término:	(1673/1777)
Tempo de Retorno:	(1673/1776)
Tempo de CPU:	(781/897)
Tempo Bloqueado por PageFault:	(142/128)
Tempo Bloqueado por Falta de Página:	(0/0)
Tempo Bloqueado por Chamada E/S:	(73/94)
Tempo Total Bloqueado:	(215/222)
Vezes Bloqueado:	(71/77)
Falha de Página:	(35/34)
Falta de Página:	(0/0)
Chamadas E/S:	(35/42)
Tempo de Espera para ser Escalonado:	(677/657)
Vezes Escalonado:	(164/165)
Número de vezes que a CPU foi perdida:	(91/89)

- Para memória de 3 paginas (7,5%):

Tempo total do Timer:	2930
Tempo Ativo de CPU:	2110
Tempo Ocioso:	820
Chamadas de Sistema:	580

Chamadas do tipo LE:	64
Chamadas do tipo GRAVA:	13
Chamadas do tipo STOP:	2
Chamadas Ilegais	0
Chamada de PageFault:	501
Trocas de Processo:	503
Número de Preempções:	22

Descrição para cada processo (0/1):

Hora de Início:	(0/1)
Hora de Término:	(2421/2930)
Tempo de Retorno:	(2421/2929)
Tempo de CPU:	(967/1143)
Tempo Bloqueado por PageFault:	(1070/1368)
Tempo Bloqueado por Falta de Página:	(44/8)
Tempo Bloqueado por Chamada E/S:	(73/94)
Tempo Total Bloqueado:	(1187/1470)
Vezes Bloqueado:	(257/323)
Falha de Página:	(221/280)
Falta de Página:	(22/4)
Chamadas E/S:	(35/42)
Tempo de Espera para ser Escalonado:	(267/316)
Vezes Escalonado:	(251/252)
Número de vezes que a CPU foi perdida:	(19/3)

Conclusão

Analisando os resultados obtidos, foi possível fazer uma análise referente a cada tipo de programa com seus respectivos processos. No caso de iniciar a memória física com espaço para todos os processos, o modelo de fila escolhido não fez diferença, uma vez que, como já dito, com espaço suficiente para todos os processos, não se fez necessário acessar a fila de prioridade para trocar as páginas de determinado endereço.

No entanto, para uma quantia de páginas suficiente para 50% das páginas

dos processos, houve uma diferença notória entre os tipos de FIFO (first in, first out) e 2ª chance. O primeiro ponto a ser analisado, é a execução mais rápida e mais eficiente no método de segunda chance. Fato pode ser observado que, no exemplo de processo em questão, é reutilizado diversas vezes uma mesma página inicial, no caso, os contadores de linhas e colunas, tais quais o endereço que guarda o número total dos mesmos. No algoritmo do tipo FIFO, esses endereços iniciais seriam os primeiros a serem substituídos em uma troca de página, causando assim, uma necessidade de realocação das páginas para proceder com o processo, enquanto no algoritmo de segunda chance, os iniciais por estarem sendo constantemente acessados, ficarão mais tempo na memória até que, se necessário, terem suas respectivas páginas liberadas. Tal observação pode ser notada numa queda de aproximadamente 30% menos chamadas de 'Page Fault' pela segundo algoritmo comparado ao primeiro, reduzindo assim, o número de bloqueios e o tempo ocioso do sistema.

Por outro lado, o terceiro teste, com o número de páginas na memória principal igual a 7,5% do necessário por todos os processos, apresentou como esperado, um dos piores resultados em questão a velocidade e eficiência, onde tantos modelos de algoritmo para a fila, obtiveram um resultado bastante parecido. Isso se dá onde, como possuíam apenas três páginas para gerenciar ambos programas, o número de realocação das páginas aumentou consideravelmente junto. Nesse exemplo, podemos ver os casos de Falta de Página, onde por exemplo, caso o processo 0 tenha alocado duas páginas para se executar uma instrução do tipo ARMM (retirar o conteúdo dado um ponteiro para um endereço), o outro processo caso queira executar uma instrução igual, será barrado e assim, terá uma interrupção para seu processo até que se possa alocar para si, as páginas desejadas. A pequena diferença do Tempo total do Timer entre ambos algoritmos, se dá pelo fato de que o algoritmo de segunda chance, naturalmente deu prioridade ao programa mais longo, uma vez que o mesmo realiza mais operações, consequentemente foi acessado mais vezes, porém essa diferença não pode ser considerada uma melhoria uma vez que para o segundo processo tivesse menos chamadas ao sistema e tempo bloqueado, o primeiro sofreu o efeito oposto, e realizou mais chamadas do tipo 'Page Fault' se comparado ao algoritmo FIFO.

Anexos

- Leitura de Json: `https://code.google.com/archive/p/json-simple/`
- Código fonte: `https://github.com/LuigiSouza/operating_system_simulator`
- Planilha com os Dados Observados: `https://docs.google.com/spreadsheets/d/1sYuPvinM7xXi9XPZ2dQPhKTIfP7htN_T8rF-mc-WHgY/edit?usp=sharing`