Universidade Federal do Ceará - Campus Quixadá
QXD0010 – Estruturas de Dados
Turma 02A – Ciência da Computação
Prof. Atílio Gomes Luiz

Projeto 1

A solução do problema descrito neste documento deve ser entregue até as 23h59min do dia **31/10/2023** pelo Moodle.

Leia atentamente as instruções abaixo.

Instruções:

- Este trabalho deve ser feito **obrigatoriamente em DUPLA** e deve ser implementado usando a linguagem de programação C++
- As duplas devem ser definidas até o dia **17/10/2023** e o nome dos integrantes deve ser enviado via Moodle no campo especificado justamente para esta tarefa.
- Até o dia **24/10/2023** deve ser submetida via Moodle uma implementação completa das estruturas de dados que serão usadas na solução do problema, assim como também deve ser enviado parte do relatório contendo a descrição detalhada das estruturas implementadas. Essa é a primeira entrega do projeto e vale nota.
- Coloque a solução do projeto em uma pasta específica. O seu trabalho deve ser compactado (.zip, .rar, etc.) e enviado para o Moodle na atividade correspondente ao Projeto 1.
- Identifique o seu código-fonte colocando o **nome** e **matrícula** dos integrantes da dupla como comentário no início de seu código.
- Indente corretamente o seu código para facilitar o entendimento. **Trabalhos com códigos maus indentados sofrerão redução na nota.**
- As estruturas de dados devem ser implementadas como TAD usando classes, como fizemos para as demais estruturas de dados programadas em aula.
- Os programas-fonte devem estar devidamente organizados e documentados.
- Observação: Lembre-se de desalocar os endereços de memória alocados quando os mesmos não forem mais ser usados.
- **Observação: Qualquer indício de plágio resultará em nota ZERO para todos os envolvidos.**

DICA: COMECE O TRABALHO O QUANTO ANTES.

1 NeoLook — Um sistema de escalonamento

Simulação é uma técnica muito utilizada para avaliação de desempenho de sistemas de computação. Um simulador cria uma abstração do sistema alvo, capturando seus aspectos mais relevantes para o estudo em questão. Neste trabalho prático, o seu objetivo é estudar o desempenho relativo de diferentes políticas de escalonamento visando o gerenciamento de recursos em um sistema distribuído chamado `NeoLook`.

1.1 Sistema Alvo

O sistema alvo de estudo é o sistema NeoLook da empresa HG, uma plataforma distribuída para armazenamento e processamento de dados. O NeoLook é composto de N computadores interligados por uma rede de alta velocidade. Cada computador contém uma CPU e dois discos.

Quando um programa (processo) é submetido para execução no NeoLook, o sistema primeiro seleciona em qual computador ele será executado. Assume-se que um processo executa inteiramente em um único computador. A execução de um processo em um computador, quando iniciada, consiste basicamente na realização de operações na CPU (processamento), seguida da realização de operações em um dos discos (recuperação de dados) do computador e, por fim, da transmissão de dados pela rede (que é compartilhada por todos os computadores). No que segue, faremos referência a estas operações executadas em cada recurso (CPU, disco ou rede) por processamento/operação/execução no recurso.

A execução de operações de um processo p em um recurso r (CPU, disco, ou rede) ocorre da seguinte maneira:

- Se o recurso r está livre (não está alocado para nenhum outro processo), p assume o recurso r, executando durante um certo intervalo de tempo D_r^p. Durante este intervalo de tempo, o recurso r é considerado alocado para p, não estando livre para demais processos que tentem acessá-lo.

- Se o recurso r não está livre (atualmente está alocado para outro processo q), p entra em uma *fila de espera* associado a r, esperando pela sua vez para executar.

- Quando p finaliza sua execução em r, p libera o recurso r, e segue tentando acessar um dos outros recursos do sistema ou finaliza sua execução. Caso existam outros processos à espera por r (isto é, na fila de espera de r), um destes processos é selecionado para executar em r. Seja este processo q; a partir deste instante, q irá executar no recurso r por um intervalo de tempo igual a D_r^q. O módulo que faz a seleção de qual processo na fila de espera de um recurso deve assumir sua execução chama-se **ESCALONADOR**.

Você pode assumir que cada processo p, ao ser direcionado para um computador do sistema para iniciar sua execução, primeiro busca realizar o processamento na CPU local. Este processamento gastará um tempo D_{cpu}^p. Após terminá-lo, o processo passa a recuperar dados de um dos dois discos do computador. Esta operação gasta um tempo D_{disco}^p. Por fim, os dados são transferidos para a rede, o que gasta um tempo D_{rede}^p.[1] Os

[1] Note que o recurso rede é compartilhado pelos processos executando em todos os N computadores do sistema (ou seja, existe um único recurso rede no NeoLook).

intervalos de tempo D^p_{cpu}, D^p_{disco} e D^p_{rede} são chamados de demandas de p pelos recursos CPU, $disco$ e $rede$, respectivamente. Após esta operação, a execução de p é considerada finalizada, e o processo deixa o sistema. Esta é uma simplificação significativa do sistema real, no qual operações de disco e de transferência pra rede são intercaladas com processamento na CPU. Porém, para fins da avaliação de desempenho pretendida, ela é razoável.

A seleção do computador no qual um dado processo deverá executar é feita de forma uniforme. Em outras palavras, qualquer um dos N computadores pode ser selecionado com igual probabilidade (no caso, com probabilidade $prob = 1/N$). Você pode considerar que o tempo necessário para realizar esta seleção é desprezível para fins da avaliação de desempenho a ser realizada.

De maneira similar, a seleção de qual dos dois discos do computador um processo deve acessar é feita de maneira uniforme: qualquer um dos dois podem ser acessados com igual probabilidade ($prob = 1/2 = 0.5$). Entretanto, um processo acessa apenas um dos dois discos. Em outra palavra, uma vez selecionado qual dos dois discos um processo p deve acessar, este executará por um intervalo de tempo igual a D^p_{disco} no disco selecionado (seguindo, ao final, para a operação na rede).

1.2 Os processos

Cada processo que executa no NeoLook pode ter valores de demandas diferentes para cada recurso (ou seja, pode ser que $D^p_{cpu} \neq D^p_{disco} \neq D^p_{rede}$). Além disto, processos diferentes podem ter demandas diferentes para um mesmo recurso ($D^p_{cpu} \neq D^q_{cpu}$). Entretanto, a demanda por CPU, D^p_{cpu}, é a mesma, independente do computador no qual ele executará. De forma similar, a demanda por disco, dada por D^p_{disco} é a mesma, independente do disco que ele acessará.

1.3 O escalonador

Conforme mencionado, o escalonador toma decisões de gerenciamento de cada fila a fim de selecionar qual processo, dentre aqueles correntemente à espera por um dado recurso r, deve iniciar sua execução em r. A fila de espera de cada recurso deve ser mantida de forma a suportar estas decisões definidas a partir de uma política de escalonamento. As seguintes políticas devem ser consideradas:

- *First Come, First Served* (FCFS): esta é uma política tradicional de gerenciamento de filas, na qual cada novo processo é sempre inserido ao final da fila, no momento da sua chegada na mesma. Em outras palavras, ele terá que esperar que o processo correntemente em execução bem como todos os que foram inseridos na fila de espera anteriormente à sua chegada terminem sua execução em r para que possa iniciar sua operação.

- *Shortest Job First* (SJF): esta política garante que o próximo processo a executar é aquele que, dentre todos na fila de espera, tenha a menor demanda pelo recurso em questão. No caso de empate, o tempo de inserção na fila deve ser considerado como fator. Ou seja, sendo dois processos com mesma demanda, aquele à espera por mais tempo têm prioridade. Esta é uma política difícil de implementar na prática pois exige conhecimento prévio sobre por quanto tempo o processo executará no recurso

(demanda), o que pode não ser conhecido e precisará ser estimado. Entretanto, pra fins de simulação, as demandas de cada processo em cada recurso serão conhecidas. Logo, a política pode ser avaliada.

1.4 O simulador

Você deverá construir um simulador do sistema NeoLook conforme especificação acima, a fim de realizar uma comparação do desempenho das duas políticas de escalonamento discutidas (FCFS, SJF). O seu simulador deve capturar os detalhes do sistema alvo bem como dos processos discutidos acima. Em outras palavras, cada um dos N computadores deverá ser representado como 3 recursos (1 CPU, 2 discos), havendo um recurso para representar a rede compartilhada. Cada recurso deverá ser representado por uma fila de espera, gerenciada conforme a política de escalonamento sendo avaliada. Portanto, o seu simulador deve implementar $3N + 1$ filas, cada uma gerenciada (através da política de escalonamento) de maneira independente. Entretanto, você pode assumir que a mesma política de escalonamento será utilizada em todos os recursos em cada simulação.

Processos são iniciados durante toda a simulação. Logo no início, um processo é atribuído a um computador (veja discussão abaixo). A partir daí, a execução deste processo será simulada através da execução do mesmo na CPU, seguida pela execução em um dos discos (selecionado conforme descrito abaixo) e por fim na rede, conforme discutido acima. Cada um destes passos deve ser simulado através da inserção/remoção do processo no respectivo recurso. Note então que o evento associado ao início da execução de um processo em um computador encadeia uma sequência de eventos, a saber: ele é imediatamente inserido na fila da CPU; eventualmente, ao término da sua execução na CPU, ele é inserido na fila de um dos discos; ao término da execução no disco, ele é inserido na fila da rede, e, ao término da execução da rede, o processo é finalizado. Este último evento corresponde ao término da execução do processo.

Para realizar a simulação, você deverá manter uma lista de eventos, cada um especificando, dentre outras coisas, um tipo (Qual o evento?) e um instante de tempo (Quando o evento ocorre?). A lista de eventos deve então ser mantida ordenada pelo instante de ocorrência, de forma a garantir que eventos são tratados em ordem cronológica. A dinâmica geral do simulador consiste então em caminhar por esta lista de eventos, realizando o tratamento adequado de cada evento no instante de ocorrência, e adiantando um relógio lógico à medida em que o tempo (simulado) progride. Este relógio nada mais é do que um contador de unidades de tempo. Todas as demandas deverão ser especificadas na mesma unidade de tempo (por exemplo: segundos).

1.5 Entrada da Simulação

A simulação deve receber como entrada dois parâmetros:

(1) uma cadeia de caracteres especificando qual política de escalonamento deve ser utilizada (FCFS, SJF);

(2) e o nome de um arquivo contendo um *trace* (lista de registros) especificando os processos cujas execuções devem ser simuladas.

A política de escalonamento especificada deve ser utilizada em todos os recursos simulados.

Cada linha do *trace* especifica a execução de um processo, seguindo o formato:
$< instante >< D_{cpu} >< D_{disk} >< D_{rede} >$

onde:

- $< instante >$ especifica o momento em que o processo é iniciado. Este instante é um valor inteiro que representa o número de unidades de tempo decorridas desde um marco inicial (instante 0);
- $< D_{cpu} >$ especifica a demanda, em unidades de tempo, de CPU do processo;
- $< D_{disk} >$ especifica a demanda, em unidades de tempo, de disco do processo;
- $< D_{rede} >$ especifica a demanda, em unidades de tempo, de rede do processo.

Um exemplo de arquivo de entrada é:

```
 0   10  20   3
10   2   15   5
12   5   50   30
...
```

O exemplo especifica o início de um processo no instante 0, com demandas na CPU, disco e rede iguais a 10, 20 e 3 (unidades de tempo), respectivamente. Um outro processo é iniciado 10 unidades de tempo depois (instante 10). Este processo tem demandas na CPU, disco e rede iguais a 2, 15 e 5, respectivamente. Duas unidades de tempo depois (instante 12), um novo processo é iniciado. Este processo tem demandas por CPU, disco e rede iguais a 5, 50 e 30, respectivamente.

O arquivo pode ter um número arbitrário de linhas (isto é, processos).

O nome do arquivo de entrada deve ser passado como parâmetro na linha de comando. Em outras palavras, o seu programa deve ser executado como, por exemplo:

```
programa FCFS arquivo_de_entrada.txt
```

onde `programa` é o nome do seu executável e `arquivo_de_entrada.txt` é o nome do arquivo com o *trace* a ser simulado.

Para tanto, você precisará utilizar primitivas para abertura, fechamento e leitura de arquivos, bem como para manipular parâmetros passados na linha de comando. Investigue o uso dos parâmetros argc e argv do main.

1.6 Saída da Simulação

O seu programa deve produzir e imprimir na tela estatísticas do desempenho do *trace* simulado. As estatísticas a serem computadas e impressas são:

- Tempo médio de execução dos processos no NeoLook: o tempo de execução de um processo inclui não somente os tempos de processamento em cada recurso mas também os tempos em que o processo ficou nas filas de espera associadas;
- Tempo médio de espera: o tempo de espera de um processo consiste na soma dos tempos de espera pela CPU, por disco e pela rede. Você deve contabilizar os tempos de espera de cada processo simulado e, ao final, computar o valor médio.
- Taxa de processamento: a taxa de processamento consiste na razão do número total de processos executados pelo intervalo de tempo decorrente desde o início do primeiro processo até o término do último. Logo, se o primeiro processo foi iniciado no instante t_1 e o n-ésimo (ultimo) foi finalizado no instante t_2, a taxa de processamento é calculada como $n/(t_2 - t_1)$.

1.7 Seleção uniforme de computadores e discos

A seleção do computador no qual um processo irá executar bem como de qual dos dois discos locais ele acessará é feita de forma probabilística seguindo uma distribuição uniforme, isto é, qualquer computador (disco) é igualmente provável de ser selecionado. Para fazer esta seleção, você precisará então fazer uso de um gerador de números pseudo-aleatórios.

Na linguagem C++, você pode usar as funções `srand48`(semente) e `drand48`(), ambas definidas em <cstdlib>. A função `srand48`(semente) é responsável por inicializar o gerador de números aleatórios e deve ser chamada no início do seu programa principal, antes de começar a simulação. O parâmetro semente é um valor inteiro qualquer. Passada uma mesma semente, você irá sempre gerar a mesma sequência de números. Logo, o parâmetro semente deve ser variado quando se quer variar os números gerados. Para fins da sua simulação, você pode usar o mesmo valor de semente sempre.

A função `drand48`() retorna um valor de ponto flutuante não negativo com precisão dupla, uniformemente distribuído no intervalo $[0, 1)$. Logo, para se gerar um número inteiro entre 0 e $N - 1$, representando a seleção de um dos N computadores, basta chamar:

```
(int)(drand48() * N)
```

Para fazer a seleção de um dos dois discos, basta então chamar (int)(`drand48`() * 2)

1.8 Implementação e Documentação

A implementação deverá ser feita utilizando alocação dinâmica de memória. Você deverá fazer vários testes com o seu programa. Para cada *trace* de entrada, o aluno deverá simular cada uma das duas políticas. O relatório deverá conter, além da descrição da estrutura geral do programa, estruturas de dados, algoritmos e ordem de complexidade de todas as funções, uma discussão do desempenho relativo das duas políticas de escalonamento em

termos das três métricas avaliadas (tempo médio de execução, tempo médio de espera, taxa de processamento). Para cada entrada testada: (1) qual política leva a um tempo de execução médio menor? (2) qual leva a uma taxa de processamento maior?

1.9 Informações adicionais

- As estruturas de dados devem ser implementadas como TAD.
- Deverá ser submetido:
  - Um relatório sobre o trabalho
  - Um pequeno manual que explique o funcionamento do sistema[2]; Esse manual pode ser uma seção do relatório;
  - Um arquivo contendo um log[3] dos testes realizados;
  - Os programas fonte devidamente organizados e documentados.
- O trabalho **deve obrigatoriamente** ser feito em dupla;
- A apresentação do trabalho será feita em horário definido pelo professor.

Obs1.: Lembrem-se de desalocar os endereços de memória alocados assim que os mesmos não forem ser mais usados.

Obs2.: Qualquer indício de plágio resultará em nota ZERO para todos os envolvidos.

DICA: COMECE O TRABALHO O QUANTO ANTES.

[2] lembrem-se de explicar como compilar e executar o programa
[3] passo a passo