Simulação de um sistema com Múltiplos servidores com perda ou atraso

Objetivos

Este trabalho tem como objetivos:

1. Obter conhecimento básico de simulação de eventos discretos usando uma abordagem prática (com o suporte de algum código parcial).
2. Análise estatística dos resultados (intervalos de confiança de 10 réplicas independentes).
3. Análise crítica dos resultados obtidos.
4. Ser capaz de usar (e construir) um Makefile simples (ao longo de todas as tarefas).
5. Escrever um programa com dois interfaces (ao longo de todas as tarefas):
   (a) Se o programa é executado sem argumentos na linha de comando, deve solicitar ao utilizador todos os parâmetros necessários ao seu bom funcionamento, fazendo controlo de erros[1], repetindo o pedido até o parâmetro digitado ter um valor válido.
   (b) O programa pode ter um argumento na linha de comando, que será o nome do ficheiro de texto (com extensão "txt") com todos os parâmetros necessários ao seu bom funcionamento. O programa lerá todos os parâmetros desse ficheiro de entrada (cujo nome foi passado como argumento na linha de comando). O formato do ficheiro deve estar descrito na sua primeira linha, a qual é lida e descartada[2]. Se houver algum erro no acesso ao ficheiro ou resultante da leitura do seu conteúdo, o programa deve terminar de forma ordeira indicando o erro ocorrido através de uma mensagem informativa.
6. Produzir código documentado usando *Doxygen* (ao longo de todas as tarefas).

Os programas (e ficheiro com resultados, quando aplicável) deverão ser submetidos no Infor-Estudante até ao prazo correspondente.

Todos os programas deverão utilizar apenas **rotinas padrão C99 ou posteriores**, ou seja, **o uso de funções disponíveis apenas num sistema operativo específico** (e.g. funções que apenas estão disponíveis no Windows) **será penalizado** a não ser que considerem também as alternativas em *linux*. Nesse caso devem incluir diretivas de compilação condicional, (e.g ver `https://www.geeksforgeeks.org/how-to-detect-operating-system-through-a-c-program/`) para o sistema operativo *linux* (e os comandos correspondentes nesse sistema). **Resumindo:** ou usam apenas código independente do sistema operativo ou devem incluir código alternativo para *linux* através de compilação condicional.

[1] O controlo de erros de digitação (e.g. letras em vez de números) será um extra, dando uma bonificação de 5% – mas a cotação final nunca será superior a 100%. Isso implicará analisar os valores devolvidos pela função *scanf*, possivelmente esvaziar o *buffer* do teclado e limpar *flags*.

[2] Se preferir pode descrever o conteúdo do ficheiro usando várias linhas, que serão descartadas.

Na descrição das tarefas que se seguem, a frase "*Na aula de **2024-??-??** espera-se que os alunos trabalhem nesta tarefa.*" não significa que a aula nessa data seja 100% dedicada ao apoio ao trabalho, mas que haverá pelo menos 50% de tempo dessa aula disponível para apoio e realização de tarefas do trabalho.

Em cada entrega, deve ser sempre submetida a versão (completa) atual do código, a sua documentação gerada pelo *Doxygen* e alguns resultados quando tal se aplique.

As datas limite são sempre sexta feira ao final do dia, com possibilidade de entrega fora de prazo sem penalização (no máximo dois dias), **exceto** na *Entrega Final*. No caso da *Entrega Final* haverá uma penalização de 5% (passa a valer 70% – só é permitido um dia de atraso).

Tarefa 1 – Data limite: 2024-10-25

Na UC Student, está disponível um programa em C para simulação por acontecimentos discretos dum sistema $M/M/1/\infty$. Estude esse programa e crie uma nova versão, após fazer as modificações necessárias para que não existam variáveis **globais** no programa. **Sugere-se** a definição de *structs* que agrupem os atributos que definem o estado do sistema, a recolha de informação estatística, a gestão do relógio e da lista de eventos, de forma a reduzir o número de argumentos das funções.

Adicionalmente modifique o programa para que as sementes das sequências pseudo-aleatórias para a geração dos tempos entre chegadas e do tempo de serviço possam ser parâmetros de entrada.

Na aula de **2024-10-23**, os estudantes devem:

1. Ser capazes de explicar o **programa em linguagem C** fornecido no livro [1] que foi disponibilizado na UC Student (com algumas pequenas alterações) e realizar as modificações necessárias para que não tenha **nenhuma variável global**.

   As sementes (índice de 1 a 100) das sequências pseudo-aleatórias para a geração dos tempos entre chegadas e do tempo de serviço devem ser lidas do ficheiro de entrada.

2. Seja organizado: liste os parâmetros de cada função sempre pela mesma ordem: primeiro, os parâmetros de saída, seguidos pelos de entrada/saída e finalmente pelos parâmetros de entrada.

3. Altere o programa principal de forma a que o nome do ficheiro com os parâmetros de entrada seja argumento da linha de comando.

   Se algum parâmetro de entrada não é válido, informe o utilizador acerca do problema detetado e termine o programa.

4. Altere o programa principal de forma a que quando este não tem argumentos na linha de entrada, os parâmetros de entrada sejam pedidos ao utilizador na consola.

   Se algum parâmetro de entrada não é válido, informe o utilizador acerca do problema detetado e repita o pedido.

5. Use o *Doxygen* para documentar a nova versão do código e criar a sua documentação. Certifique-se que deixa claro quais são os parâmetros de saída e os parâmetros de entrada (ou ambos).

A entrega, confirmando a conclusão desta tarefa, será feita no *InforEstudante*, através da submissão do código (e sua documentação criada usando o Doxygen) e de um *Makefile* simples para compilá-lo.

Na aula de **2023-10-23** espera-se que os alunos trabalhem nesta tarefa.

Tempo para completar esta tarefa: 1 semana.

Tarefa 2 – Data limite: 2024-11-08

Crie uma nova versão (salve a anterior), onde os novos parâmetros de entrada definem:

- O número de servidores (≥ 1).

  O que implica alterar o estado do sistema e a lista de eventos.

- Se o sistema é com espera (fila infinita) ou com perda (sem fila).

  O que implica ter um parâmetro adicional, que indica qual o tipo de sistema a simular, e novas variáveis estatísticas.

Ou seja, o seu código deverá ser capaz de simular um sistema $M/M/n/\infty$ (Erlang-C) ou $M/M/n/0$ (Erlang-B).

Note que precisará de sementes diferentes para cada sequência pseudo-aleatória:

1. Geração dos tempos entre chegadas (depende da intensidade do processo de chegadas).
2. Seleção do servidor a utilizar entre os servidores livres.

   A seleção sequencial não é a mais adequada, pois sobrecarrega de forma diferente os vários servidores, contudo conduz a um código mais simples e os resultados continuam válidos do ponto de vista da qualidade de serviço oferecida (atraso médio ou probabilidade de bloqueio, conforme o sistema).

   Uma implementação mais realista (que requer mais uma sequência de números pseudo-aleatórios) será valorizada com 5 pontos extra.
3. Geração dos tempos de serviço em cada servidor (depende do tempo médio de serviço em cada servidor, considerado igual nos n servidores).

   Note que para garantir a independência estatística entre as ocupações nos vários servidores, será necessário utilizar uma sequência de números pseudo-aleatórios para cada um dos servidores.

Nas aulas de **2024-10-30** e **2024-11-06** espera-se que os alunos trabalhem nesta tarefa.

Tempo para completar esta tarefa: 2 semanas.

Tarefa 3 – Data limite: 2024-11-15

Crie uma nova versão em que:

- Otimiza o código tornando a fila de espera circular (para não precisar de deslocar todos os elementos sempre que remove um utilizador do início da fila).

  Pode utilizar a implementação de **fila circular fornecida**, para gerir uma fila de espera **FIFO**. Esta implementação utiliza uma estrutura cuja dimensão pode ser alargada (reserva espaço usando *malloc* e quando o espaço reservado se esgota, alarga-o usando *realloc*).

- Considere como parâmetro adicional a disciplina da fila de espera: (i) o primeiro a chegar é o primeiro a ser servido (**FIFO** – *First In First Out*); (ii) o último a chegar é o primeiro a ser servido (**LIFO** – *Last In First Out*).

- Adapte o código fornecido para gerir a fila circular (com disciplina FIFO), e construa uma nova *struct LIFO* e implemente os métodos necessários.

  Questão: Esta nova fila precisa de ser circular?

Garanta que recolhe em ficheiro, além de todos os parâmetros de entrada (para boa identificação da experiência), os valores de saída relevantes, no formato adequado a tratamento posterior:

(a) número médio de servidores ocupados;

(b) número médio de utilizadores à espera;

(c) tempo médio de espera;

(d) tempo médio de espera para utilizadores que esperaram;

(e) no caso de sistemas com perda, determine o bloqueio médio de pedidos de serviço;

(f) taxa de ocupação de cada servidor.

(g) taxa de ocupação dos servidores (em conjunto) - pode ser obtida a partir da medida anterior.

Na aula de **2024-11-13** espera-se que os alunos trabalhem nesta tarefa.

Tempo para completar esta tarefa: 1 semana.

Tarefa 4 – Data limite: 2024-11-22

Escreva um programa principal (função *main*) como descrito no Item 5 na Página 1, caso não o tenha feito ainda.

Considere que o nome do ficheiro com os resultados de saída, ou é o segundo argumento na linha de comando ou é criado com base no nome do ficheiro com os parâmetros de entrada.

Note que esta tarefa apenas assegura a *organização* do trabalho realizado até aqui, caso não o tenha feito ainda longo das tarefas 1 a 3!

Inclua alguns exemplos de execução e resultados obtidos (para sistemas com atraso e perda) com 1 ou mais servidores.

Sugestão 1: Considere, caso não o tenha feito ainda, que faz parte dos parâmetros de entrada a indicação das sequências pseudo-aleatórias a utilizar (1 a 100).

Sugestão 2: Considere durações crescentes para as simulações realizadas e analise os resultados.

A versão entregue deverá vir acompanhada dum relatório final produzido pelo seu programa para cada um dos seguintes sistemas:

- $M/M/1/\infty$
- $M/M/n/\infty$, com n à sua escolha no intervalo $[3, 7]$
- $M/M/1$
- $M/M/n$, com n à sua escolha no intervalo $[3, 7]$

Cada um desses relatórios deverá conter a informação que foi extraída do ficheiro de entrada, para que seja possível saber as condições subjancentes aos resultados obtidos·

Na aula de **2024-11-20** espera-se que os alunos trabalhem nesta tarefa.

Tempo para completar esta tarefa: 1 semana.

Tarefa 5 – Data limite: 2024-12-06

Considere 10 execuções independentes e calcule intervalos de confiança (IC) de 95% para as médias de interesse estimadas obtidas.

Terá de decidir qual a duração da corrida de simulação.

Lembre-se de que cada execução deve usar sementes totalmente diferentes para a geração de números aleatórios. Em cada corrida de simulação precisará de $1+n$ sementes: 1 para o processo de chegadas e as restantes para cada um dos processos de saída dos n servidores (ou $2 + n$ se escolher aleatoriamente o servidor a ocupar, entre os que se encontram livres). Isto implica que o número n de servidores é limitado a 9 ou 8, respetivamente.

Para esta tarefa, é necessária uma nova função de preparação dos dados para posterior tratamento estatístico. Sugere-se que os resultados sejam armazenados como um arquivo CSV que pode ser facilmente importado para alguma folha de cálculo. Mas existem outras abordagens alternativas que pode explorar para obter o IC como escrever um programa adicional para tratamento estatístico dos resultados.

Compare os valores de simulação obtidos (incluindo o CI) com resultados exatos obtidos para o estado estacionário de multi-servidor com atraso (sistema Erlang-C) nas mesmas condições, e também para um sistema multi-servidor com perdas (Erlang-B), considerando diferentes valores para o número de servidores, usando resultados obtidos com a implementação das fórmulas dos sistemas de Erlang-C e Erlang-B (respetivamente) feitas com apoio em sala de aula.

Sabemos que para um sistema multi-servidor com atraso (sistema Erlang-C) – $M/M/n/\infty$:

$$\overline{W} \;=\; \overline{W}_e \cdot P[W > 0] \tag{1}$$

$$\overline{W}_e = E\{W|W>0\} = \frac{h}{n-A} \quad (2)$$

$$E_c(A,n) = P[W>0] = \frac{nE_B(A,n)}{n-A(1-E_B(A,n))} \quad (3)$$

$$L_q = \lambda\, E_c(A,n)\, \overline{W}_e \quad (4)$$

onde W é a variável aleatória que representa o tempo de espera dos clientes na fila, com média $\overline{W}$; W_e é a variável aleatória que representa o tempo de espera dos clientes que esperaram mais que zero $(W|W>0)$, cuja média é $\overline{W}_e$; $1/\lambda$ é o tempo médio entre chegadas; h é o tempo médio de serviço (igual para todos os n servidores); $A=\lambda h$ é o tráfego médio oferecido; e L_q é o número médio de clientes na fila.

Nesta tarefa devem ser submetidos, além do código desenvolvido até ao momento, os ficheiros de saída e o resultado do respetivo tratamento estatístico (não se esqueça de analisar os resultados das disciplinas FIFO e LIFO). Espera-se que os alunos façam uma análise crítica dos resultados obtidos.

Na aula de **2024-12-04** espera-se que os alunos trabalhem nesta tarefa.

Tempo para completar esta tarefa: 1 semana são duas semanas de calendário mas apenas uma de trabalho, devido à realização da frequência, prevista para 27 de novembro).

Entrega final – Data limite: 2024-12-11

O pacote de entrega final: o código C final, um ficheiro *Makefile* simples para compilação, a documentação produzida com *Doxigen* e a análise dos resultados da Tarefa 5.

Note que esta tarefa dá-lhe a oportunidade de completar/corrigir a entrega da Tarefa 5!

Na aula de **2024-12-11** espera-se que os alunos esclareçam quaisquer dúvidas finais referentes ao seu trabalho.

Por fim, no dia 13 de dezembro, os alunos deverão estar preparados para defender o seu trabalho, em horários a ser disponibilizados no Nónio (defesa de trabalhos) em intervalo a definir.

Avaliação

A entrega de cada tarefa será creditada com 5 pontos conforme tabela abaixo. Nenhuma avaliação formal será feita até à *Entrega Final*, mas serão realizadas algumas verificações para confirmar se o material entregue corresponde à Tarefa a ser concluída em cada data. Os alunos deverão aproveitar o tempo presencial para esclarecer quaisquer dúvidas relativas aos trabalhos em curso.

Tarefa	Data Limite	Pontos (0-100) %
Tarefa 1	2024-10-25	5
Tarefa 2	2024-11-08	5
Tarefa 3	2024-11-15	5
Tarefa 4	2024-11-22	5
Tarefa 5	2023-12-06	5
Entrega final	2024-12-11	75

A **Entrega Final**, em caso de atraso de um dia, sofrerá uma penalização, limitando o valor daquela componente a 70%. Nenhuma entrega será aceite após 12 de dezembro.

No dia **13 de dezembro**, os alunos deverão estar preparados para **defender** o seu trabalho, para o que se deverão inscrever no sistema *InforEstudante*. O não comparecimento nessa data resultará em 0 pontos para todo o trabalho. Caso a ausência seja justificada, deverá ser planeada (preferencialmente) com antecedência uma data alternativa para a defesa do trabalho. A nota final será o produto da nota da defesa pela nota do trabalho (entregas parcelares e avaliação da entrega final).

Referências

[1] Averill M. Law.*Simulation Modeling and Analysis*, McGraw-Hill Series in Industrial Engineering and Management, 5th Edition, 2015.