Sétimo trabalho:
Simulação de um sistema computacional

(Vale 20 pontos)

2016-06-13

1 A simulação

Um sistema computacional com P processadores deve executar um conjunto de N tarefas, cada uma delas demandando um tempo total de processador de T_i, para $0 \leq i \leq N-1$.

Os processadores funcionam num esquema de *time-sharing*, onde uma tarefa faz uso de um processador por um tempo q (o *quantum*) e depois o processador é entregue a outra tarefa que esteja esperando. Se a tarefa não terminou durante a execução de seu último quantum, ela é inserida novamente no final da fila de tarefas aguardando para executar (de acordo com a política do escalonador, veja adiante). A entidade responsável por determinar qual a tarefa que deve ser executada quando um processador fica disponível é chamada *escalonador*. A troca de uma tarefa para outra demora s (tempo de escalonamento). Devido a características internas do sistema, sempre que uma tarefa for executar em um processador diferente daquele em que executou da última vez (e também quando for executar pela primeira vez em qualquer processador), ela incorre numa penalidade de troca de contexto, denominada por c (em unidades de tempo).

Resumindo, quando um processador fica disponível ele avisa o escalonado, que gasta um tempo s para determinar qual a próxima tarefa a executar e a envia ao processador, que atribui q instantes para sua execução. Se for a primeira execução da tarefa ou se for uma execução em processador diferente do anterior, um total de c (do tempo q originalmente alocado para a tarefa) será desperdiçado, e estarão disponíveis para a tarefa apenas $q-c$ unidades de tempo. Suponha que o tempo restante para execução da tarefa seja t. Neste caso, se $t > q$ (ou $t > q - c$ se houver penalidade), então o tempo restante deve ser correspondentemente atualizado. Caso $t \leq q$ (ou $t \leq q - c$), então considera-se que a tarefa terminou após t unidades de tempo, e processador fica disponível mais cedo.

O simulador deverá implementar dois tipos de escalonadores distintos. No primeiro, há apenas uma fila de tarefas esperando e quando um processador fica disponível, a tarefa no topo da fila é enviada para execução nele (sem levar em consideração onde ela havia sido executada pela última vez). No segundo, há uma fila para cada processador, e quando um deles fica disponível ele executa a primeira tarefa na fila correspondente. Neste último caso, a tarefa é encaminhada aleatoriamente a uma das filas no momento em que chega para execução.

Você deve criar um simulador para esse sistema usando o módulo de simulação por eventos discretos apresentado em aula. Para simular os valores de T_i, você irá utilizar uma distribuição log-normal, com o parâmetro de localização variável e parâmetro de escala igual a 1.0. Para sortear números com essa distribuição, use a função `random.lognormvariate(loc, 1.0)`, onde `loc` é o parâmetro de localização. Em diferentes simulações, varie o parâmetro de localização na faixa $[1, 5]$. Cada tarefa chega para execução no sistema em um instante distinto. Vamos simular os tempos de chegada da seguinte forma: A primeira tarefa sempre chegará no instante 0; depois, cada tarefa seguinte chegara com um intervalo da tarefa anterior que será determinado por uma distribuição exponencial, isto é, sorteamos um valor usando `random.expovariate(taxa)` (onde `taxa` é a taxa de chegada de tarefas) e o somamos ao tempo de chegada da tarefa anterior para determinar o tempo de chegada da nova. Em simulações diferentes, use valores de `taxa` na faixa $[0.1, 1.5]$.

Para cada tarefa i, serão registrados os tempos de chegada e de término; com esses valores, calculamos o seu *tempo extra*, que corresponde à diferença entre o tempo para completar a execução e o tempo T_i da tarefa. Para cada processador, deve ser registrado o tempo total que ele ficou desocupado

(terminou de executar uma tarefa e ainda não recebeu a próxima). Para a simulação como um todo, queremos saber o tempo total desde o início até o instante em que a última tarefa terminou de executar.

2 Os estudos a realizar

Com o simulador desenvolvido, você vai fazer os seguintes estudos:

1. Usando o escalonador com fila única, fixe o número de tarefas $N = 1000$, o número de processadores em $P = 2$, o parâmetro de localização da log-normal em 2 e a taxa de chegada de tarefas (parâmetro da exponencial) em 0.25. Use $q = 1$, $s = 0.1$ e $c = 0.3$. Execute a simulação 10 vezes com esses mesmos parâmetros e mostre: a média e desvio padrão do tempo total de execução; um histograma dos tempos extras das tarefas e um histograma dos tempos ociosos dos processadores.

2. Repita o mesmo estudo anterior, mas com o escalador que tem uma fila por processador. Compare os resultados.

3. Repita os estudos dos dois itens anteriores para $P = 1$ e $P = 4$.

4. Usando $N = 1000$, $P = 2$, $q = 1$, $s = 0.1$, $c = 0.3$ e o parâmetro da log-normal em 2, varie o parâmetro da exponencial na faixa $[0.1, 1.5]$ e levante um gráfico do tempo total de execução em função desse parâmetro para os dois tipos de escalonadores. Compare os resultados.

5. Usando $N = 1000$, $P = 2$, $q = 1$, $s = 0.1$, $c = 0.3$ e o parâmetro da exponencial em 0.25, varie o parâmetro da log-normal na faixa $[1, 5]$ e levante um gráfico do tempo total de execução em função desse parâmetro para os dois tipos de escalonadores. Compare os resultados.

6. Usando $N = 1000$, $P = 2$, $q = 1$, $s = 0.1$, o parâmetro da log-normal em 2 e o parãmetro da exponencial em 0.25, varie c na faixa $[0.05, 0.5]$ para os dois escalonadores e compare a influência desse parâmetro em cada um deles.

7. Usando $N = 1000$, $P = 2$, $q = 1$, $c = 0.3$, o parâmetro da log-normal em 2 e o parãmetro da exponencial em 0.25, varie s na faixa $[0.02, 0.5]$ para os dois escalonadores e compare a influência desse parâmetro em cada um deles.

8. Por fim, considerando todos os resultados anteriores, pode-se concluir que, dentro da faixa de valores estudados, um dos escalonadores é sempre melhor do que o outro? Se não, em que situações cada um deles é preferível?

3 A entrega

O trabalho deverá ser feito em grupos de dois alunos.

A entrega deve ser feita até o dia 2016-07-02, às 23:55, pelo STOA. Entreguem um notebook com o código, os resultados das simulações e as discussões.

O seu simulador deve usar o módulo `desimul.py` (apresentado em aula) sem alteração.