Overcooked com Programação Concorrente

Wladimir Ganzelevitch Mesquita Gramacho - 15/0048718
Departamento de Ciência da Computação - CIC/UnB
Programação Concorrente - Prof. Eduardo Alchieri

Novembro de 2018

1 Introdução

Durante o curso de Programação Concorrente, aprendemos a utilizar *threads* e compartilhamento de memória para resolver problemas onde dois ou mais processos competem pelo mesmo recurso. Dessa forma, solucionamos problemas de forma mais eficiente ao executar código no mesmo tempo lógico [1][2].

Este trabalho tem o objetivo de colocar em prática esse conhecimento, solucionando um problema proposto pelo aluno. Como especificado, a solução deve utilizar a biblioteca POSIX Pthreads[1] da linguagem C para controlar o acesso às regiões críticas dos processos. Com isso, temos *locks mutex*, *locks* condicionais, semáforos e barreiras à nossa disposição.

O problema proposto neste trabalho se trata de uma simulação do jogo Overcooked[2] onde jogadores cooperam numa cozinha, por vezes em locais bastante adversos, para cozinhar o máximo de pratos possível. Ao desenvolver a solução, foi necessário alterar um pouco a lógica do jogo para que pudéssemos utilizar concorrência. Por fim, concluímos este relatório trazendo aspectos do desenvolvimento da solução e possíveis próximos passos.

2 Formalização do Problema

Como já dito antes, o problema proposto foi o de fazer uma simulação do jogo Overcooked. A ideia do problema é fazer com que os jogadores usem a cozinha o mais eficiente possível, utilizando tábuas de cortar alimentos e frigideiras. Na Figura 1, podemos ver uma captura de tela do jogo, na qual quatro jogadores estão na no meio de uma rua cozinhando sopas de tomate e de cebola. No

[1] https://computing.llnl.gov/tutorials/pthreads/
[2] https://en.wikipedia.org/wiki/Overcooked

jogo, a melhor estratégia para fazer mais pontos é coordenar os jogadores para que cada um foque em somente uma atividade, de modo que um não atrapalhe o outro ao tentar utilizar o mesmo instrumento. Além disso, essa estratégia possibilita que o jogador não perca tempo ao andar no meio da cozinha, o que poderia prejudicar ele e a equipe inteira.

Figura 1: Captura de tela do jogo Overcooked

Por outro lado, se utilizarmos essa estratégia de dividir para conquistar no problema, não teremos um problema intuitivamente concorrente, pois os jogadores não irão concorrer pelos instrumentos (recursos) da cozinha. Desse modo, faz-se necessário utilizar uma abordagem mais competitiva para que utilizemos processos concorrentes. É interessante também criar uma escassez de recursos, diminuindo a quantidade de tábuas para cortar alimentos para 2 tábuas e a quantidade de frigideiras para 1, de modo que os jogadores teriam que, literalmente, concorrer pelos instrumentos da cozinha. Dito isso, podemos avançar para a solução do nosso problema.

3 Solução

3.1 Ideia da Solução

No nosso problema, os jogadores concorrem por instrumentos da cozinha. Além disso, os jogadores também teriam que disputar pelo ingrediente que viria através de uma esteira de alimentos. Essa esteira de alimentos "produziria"um alimento de tempos em tempos e este ficaria lá até um jogador o pegasse. A esteira tem tamanho infinito, ou seja, vai suportar qualquer quantidade de alimentos.

Desse modo, já podemos determinar o fluxo de trabalho de um jogador na cozinha:

1. Pega ingrediente na esteira;
2. Corta o ingrediente em uma das duas tábuas de cortar;
3. Usa a frigideira para cozinhar o ingrediente;
4. Entrega o prato pronto.

No momento que esse jogador entrega um prato, ele incrementa a contagem de pontos que tem. Nesse momento, o ranking deve ser exibido. Na Figura 2 podemos ver um esquema da solução. Nesse caso, os jogadores 1 e 2 estão esperando um ingrediente na esteira. O jogador 3 está cortando o seu ingrediente enquanto o jogador 4 já está na posse da frigideira. No momento que o jogador 4 terminar de usar a frigideira e entregar seu prato, ele entra novamente na disputa por um novo ingrediente.

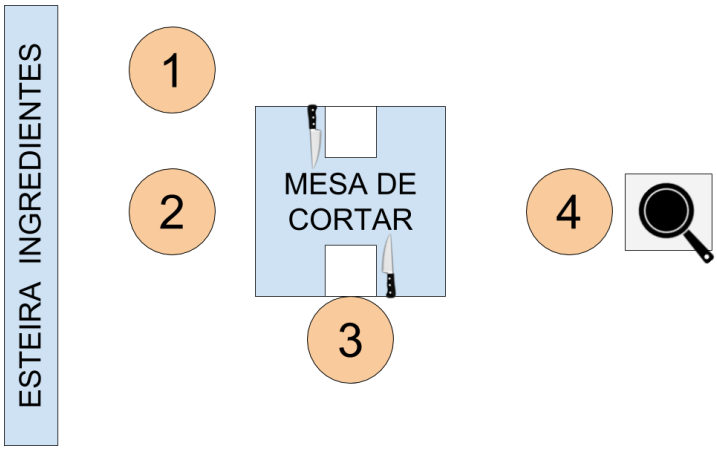

Figura 2: Esquema da solução

3.2 Desenvolvimento da Solução

Definida a ideia da solução, podemos ir para o seu desenvolvimento. Pela especificação do trabalho, tínhamos que utilizar a biblioteca POSIX Pthreads. Essa biblioteca está disponível tanto para a linguagem C como para C++, mas optamos por usar a primeira porque a solução não precisa das funcionalidades extras que o C++ tem em relação ao C.

Para desenvolver uma solução que faça jus ao esquema acima, precisamos de 5 threads: 4 para cada jogador e uma para a esteira de alimentos. O código da esteira de alimentos é simples: dentro de um loop infinito, produz mais um ingrediente. Para evitar condições de corrida com os jogadores, é necessário criar

um *lock* para os ingredientes na esteira. Além disso, faz-se necessário ter um *lock* condicional para "avisar"os jogadores que há um ingrediente na esteira. É interessante também colocar uma espera para que a esteira não faça ingredientes continuamente.

Com a esteira pronta, podemos partir para o código mais complexo dos jogadores. Como já dito, um jogador pega um ingrediente, corta, cozinha e entregue um prato pronto. Dessa forma, podemos dividir seu trabalho em quatro funções.

Na primeira, o jogador deve conseguir um ingrediente. Para isso, ele deve pegar o *lock* e avaliar se há ingredientes disponíveis na esteira. Se não houver, ele deve ficar preso num *lock* condicional que só será aberto quando a *thread* da esteira adicionar um ingrediente. Quando isso acontecer, o jogador volta a avaliar se há ingrediente ou se algum outro jogador pegou antes dele. Se ele tiver pego o ingrediente, ele decrementa a quantidade de ingredientes disponíveis e parte para a mesa de cortar.

Na mesa de cortar, o jogador tem que pegar uma permissão de um semáforo da mesa de corte. Esse semáforo é inicializado com 2 permissões, pois temos 2 tábuas de cortar. Ao conseguir pegar uma permissão, ele corta seu ingrediente. Ao finalizar, ele esperará para ter uma oportunidade de pegar a frigideira. Para não ficar no meio da cozinha com o ingrediente cortado na mão, ele só sai da mesa de cortar quando conseguir pegar a frigideira.

Ao pegar a frigideira, ele libera a permissão da tábua de cortar. Ao terminar de cozinhar seu prato, ele libera a frigideira, entrega seu prato e pontua no jogo. Por fim, o jogador volta à esteira para pegar um novo ingrediente. Na Figura 3 podemos ver uma execução da nossa solução.

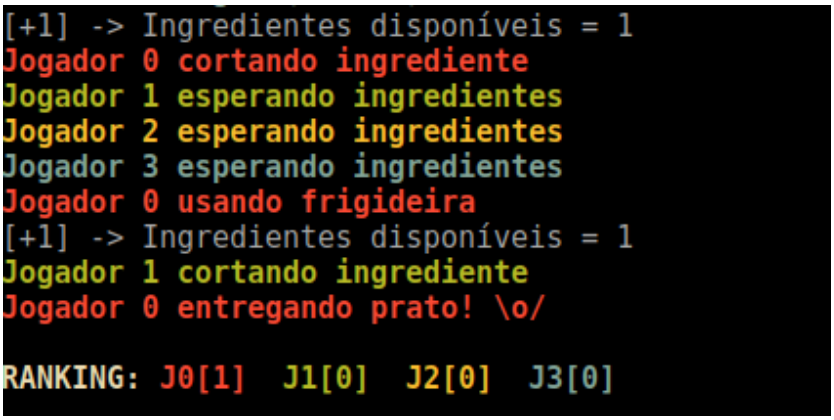

Figura 3: Execução da Solução

4 Conclusão

No desenvolvimento dessa solução, foi necessário a utilização de diferentes estruturas de controle de *threads*. Com isso, conseguimos atingir o objetivo deste trabalho que era de colocar em prática o que foi aprendido em sala de aula e gerenciar processos concorrentes com o intuito de torná-los mais eficientes. Como próximos passos para este trabalho, podemos fazer uma interface gráfica para termos uma noção visual do que cada jogador está fazendo. Uma das ideias é utilizar a biblioteca ncurses[3] para fazer essa interface gráfica.

Referências

[1] Wikipedia contributors, Concurrent computing — Wikipedia, The Free Encyclopedia, 2018 `https://en.wikipedia.org/w/index.php?title=Concurrent_computing&oldid=855021401`, [Online; accessed 4-November-2018]

[2] Eduardo Alchieri, Introdução à Programação Concorrente, `https://cic.unb.br/~alchieri/disciplinas/graduacao/pc/introducao.pdf`, [Online; accessed 5-November-2018]

[3] https://linux.die.net/man/3/ncurses