Relatório trabalho 2 - CI128

Gabriel Silva Hermida
Vinícius Teixeira Vieira dos Santos

July 2021

1 Overview

Nosso segundo trabalho da disciplina é composto por 6 arquivos. Todos, exceto pelo "main.c", têm seus arquivos cabeçalho. São eles: "erro", "graph", "log", "output" e "equivalente". O porquê de cada um deles:

1. "erro": Contém a função que finaliza a execução do programa caso algum caso não previsto aconteça.
2. "graph": Contém as estruturas de dados utilizadas para representar e armazenar os vértices de um grafo, criado pelo algoritmo 1, além de funções que manipulam essas estruturas.
3. "log": Contém estruturas de dados utilizadas para representar e armazenar as transações lidas pela entrada em forma de uma 'log'. Foi utilizada para confirmar se a leitura estava correta e serve para a construção das visões propostas pelo algoritmo 2. Há uma função que imp
4. "output": Contém as funções responsáveis por gerar a saída do programa.
5. "equivalente": Contém as funções responsáveis por gerar visões do agendamento dado e testar a equivalência delas, conforme o funcionamento do algoritmo 2.

2 Descrição do funcionamento

Nosso programa funciona seguindo o seguinte fluxo: Enquanto lê as operações da entrada que sejam diferentes de "commit", cria vértices para cada novo índice de transação lido e cria a estrutura de log com base em todas as informações fornecidas por uma transação. Sempre, após uma nova leitura, verifica-se as condições de conflito entre ela e as anteriores (condições listadas pelo algoritmo 1), e caso um conflito seja localizado, um arco é criado entre os vértices relativos às operações conflitantes. Ao encontrar um commit, o vértice equivalente à operação "commitada" é marcado como "commitado", e é feita a uma verificação: se todos os vértices estiverem marcados como "commitados", procede-se

para a busca do ciclo no grafo criado durante a execução (final do algoritmo 1) e criam-se as visões para comparar a equivalência entre elas (algoritmo 2). Dadas as informações de retorno dessas duas funções, a saída é gerada, as listas de logs e de vértices são esvaziadas e o laço se repete.

3 Particularidades

3.1 Detecção de ciclo em grafo

Para a detecção de um ciclo em um grafo, previsto pelo algoritmo 1, as funções foram implementada à mão. O processo é composto por duas partes: iterar pelos vértices (na função "verifica_ciclo") criados pela leitura do agendamento e, a partir de um vértice selecionado (conforme a ordem de aparição deles), recursivamente iterar pela sua vizinhança (dados os arcos criados com base nos conflitos encontrados), incrementando a variável "visitado" ao "entrar" e "sair" de um vértice (função "itera_adjascentes"). É considerado que um grafo tem um ciclo se a iteração pela sua vizinhança "entrou" em um vértice cujo valor de "visitado" é igual a 1. Os vértices cujo valor de "visitado" seja 2 serão desconisderados na continuidade da busca pelo ciclo.

3.2 Visão equivalente

Para a criação da visão equivalente (função 'equivalente'), conta-se a quantidade de transações que estão sendo envolvidas no agendamento e cria-se um vetor com isso. Atribui-se cada um dos índices das transações à uma posição do vetor. A partir daí, as posições do vetor sofrem permutações conforme o Algoritmo de Heap (função generate) e as visões são geradas (função criaAgendamento) com base na ordem em que as transações aparecem, para que sejam comparadas à original (função comparaAgendamentos) e tenham sua equivalência validada.