Bài tập lớn 1
Hiện thực Cache

Tháng 7/2022

1 Chuẩn đầu ra

Sau khi hoàn thành bài tập lớn này, sinh viên sẽ có khả năng

- Giải thích được cơ chế hoạt động của cache.
- Ứng dụng danh sách liên kết đơn trong vấn đề lưu trữ dữ liệu trong một cache đơn giản.
- Ứng dụng cây BST vào vấn đề tìm kiếm trong một cache đơn giản.

2 Giới thiệu

Trong một máy tính thực, tốc độ của bộ nhớ chính rất thấp so với tốc độ của bộ vi xử lý của nó. Để cải thiện hiệu suất, bộ nhớ cache có tốc độ cao được sử dụng để giảm thời gian chờ của bộ vi xử lý khi truy cập các tập lệnh và dữ liệu trong bộ nhớ chính. Cache lưu trữ các bản sao của dữ liệu từ bộ nhớ chính để các yêu cầu liên quan đến những dữ liệu đó trong tương lai có thể được đáp ứng nhanh hơn.

2.1 Cơ chế hoạt động của cache

Khi vi xử lý lần đầu tiên tham chiếu đến một ô nhớ trong bộ nhớ chính, nội dung của ô nhớ cùng với địa chỉ của nó sẽ được đưa vào cache. Sau đó, khi vi xử lý tham chiếu lại đến cùng một ô nhớ, nội dung mong muốn sẽ được đọc trực tiếp từ cache tốc độ cao thay vì bộ nhớ chính có tốc độ chậm hơn.

2.2 Chính sách thay thế cache

Vì kích thước của cache là có giới hạn, khi cache đầy và một dữ liệu mới được thêm vào cache cho các yêu cầu trong tương lai, một vài dữ liệu hiện có trong cache phải được lựa chọn theo một chính sách nào đó để xóa đi và nhường chỗ cho dữ liệu mới. Có nhiều chính sách thay thế bộ nhớ cache, một vài trong số đó sẽ được liệt kê và yêu cầu hiện thực trong bài tập lớn này. Cụ thể:

- FIFO (First In First Out): dữ liệu cũ nhất trong cache sẽ được chọn.
- LIFO (Last In First Out): dữ liệu mới nhất trong cache sẽ được chọn.

3 Yêu cầu của bài tập lớn

3.1 Mô tả

Bài tập lớn này yêu cầu hiện thực một cache để truy cập dữ liệu (đọc/ghi).

3.1.1 Giao diện cache

Bài tập lớn phải được hiện thực dưới dạng một lớp Cache với những phương thức sau:

- **Data* read(int addr)**: trả về dữ liệu được lưu tại địa chỉ **addr** nếu **addr** được lưu tại cache, ngược lại trả về NULL. Kiểu dữ liệu Data sẽ được mô tả sau.
- **Elem* put(int addr, Data* data)**: đưa **addr** và **data** vào trong cache và trả về phần tử bị đưa ra khỏi cache nếu có, ngược lại trả về NULL.
- **Elem* write(int addr, Data* cont)**: tìm kiếm nếu **addr** có ở trong cache. Nếu có, phương thức này sẽ thay thế dữ liệu liên kết với **addr** với **cont**. Ngược lại nếu không tìm thấy, phương thức đưa **addr** và **cont** vào trong cache. Phương thức này trả về phần tử bị đưa ra khỏi cache nếu có, ngược lại trả về NULL. Lưu ý rằng **cont** cùng với địa chỉ **addr** của nó được đưa vào cache, và không phải vào bộ nhớ chính, vì vậy dữ liệu trong cache lúc này không còn là bản sao của dữ liệu trong bộ nhớ chính nữa. Do đó, thuộc tính **sync** của phần tử trong cache phải được đặt thành false.
- **void print()**: in giá trị của các phần tử trong cache theo thứ tự tăng dần thời gian sống của mỗi phần tử. Mỗi phần tử phải được in nội dung của nó bằng phương thức **print** của lớp Elem.
- **void preOrder()**: in giá trị của các phần tử trong cache theo tiền thứ tự của cây BST, cây được sử dụng để tìm kiếm một địa chỉ trong cache.
- **void inOrder()**: in giá trị của các phần tử trong cache theo trung thứ tự của cây BST.

Khi một phần tử mới được đưa vào **full** cache (thông qua phương thức **put** hoặc **write**), chính sách thay thế **FIFO** (dùng khi addr có giá trị chẵn) và **LIFO** (dùng khi addr có giá trị lẽ) sẽ được sử dụng để xóa bỏ một phần tử đang nằm trong cache. Lưu ý rằng kích thước của cache được cung cấp thông qua **MAXSIZE** trong file "main.h". Để tìm kiếm trong cache, một cây BST với khóa là địa chỉ phải được hiện thực.

3.1.2 Giải thích các phương thức

Khi vi xử lý muốn đọc dữ liệu tại địa chỉ **addr**, nó sẽ gọi phương thức **read(addr)** của lớp Cache. Nếu địa chỉ **addr** cùng với nội dung của nó nằm trong cache, nội dung sẽ được trả về. Ngược lại, NULL sẽ được trả về và vi xử lý sẽ đọc dữ liệu **cont** từ bộ nhớ chính và gọi **put(addr, cont)** để đưa (addr, cont) vào cache. Khi vi xử lý muốn ghi dữ liệu **cont** vào địa chỉ **addr**, nó sẽ gọi phương thức **write(addr, cont)**. Phương pháp này sẽ thay đổi nội dung của địa chỉ **addr** nếu có, hoặc thêm một mục mới vào cache có **sync** là false. Khi phương thức **put** hoặc **write** trả về một mục (do cache đã đầy), vi xử lý sẽ kiểm tra **sync** của mục đó. Nếu là false, vi xử lý sẽ ghi dữ liệu của mục vào bộ nhớ chính.
Các phương thức khác của lớp Cache được sử dụng chỉ để kiểm tra cách hiện thực của sinh viên.

3.1.3 Dữ liệu đầu vào và đầu ra

Các lớp dưới đây được định nghĩa trong file "main.h" và được sử dụng như dữ liệu đầu vào và đầu ra:

- Lớp Float: sử dụng cho kiểu dữ liệu số thực
- Lớp Int: sử dụng cho kiểu dữ liệu số nguyên
- Lớp Bool: sử dụng cho kiểu dữ liệu luận lý
- Lớp Address: sử dụng cho nội dung địa chỉ
- Lớp Data: sử dụng để biểu diễn bất kì dữ liệu nào
- Lớp Elem: sử dụng một phần tử trong cache với 3 trường chính:
  - **addr**: lưu trữ địa chỉ của ô nhớ.
  - **data**: lưu trữ nội dung của ô nhớ.
  - **sync**: cho biết nội dung trong cache có giống với nội dung của ô nhớ tương ứng trong bộ nhớ chính hay không.

3.2 Instructions

Để hoàn thành bài tập lớn này, sinh viên phải:

- Tải xuống tập tin initial.zip và giải nén nó
- Sau khi giải nén sẽ được 4 files: main.cpp, main.h, Cache.cpp và Cache.h. Sinh viên KHÔNG ĐƯỢC sửa đổi các file main.cpp, main.h
- Sửa đổi các file Cache.h và Cache.cpp để hiện thực cache.
- Đảm bảo rằng chỉ có một lệnh **include** trong file Cache.h là **♯include "main.h"** và một **include** trong file Cache.cpp đó là **♯include "Cache.h"**. Ngoài ra, không cho phép có một include nào khác trong các file này

4 Nộp bài

Sinh viên chỉ nộp 2 files: Cache.h và Cache.cpp, trước thời hạn được đưa ra trong site "**Cấu trúc dữ liệu và giải thuật (CO2003)_HK213_ALL**". Sinh viên có thể nộp bài bao nhiêu lần tùy ý nhưng chỉ có bài nộp cuối cùng được tính điểm. Vì hệ thống không thể chịu tải khi quá nhiều sinh viên nộp bài cùng một lúc, vì vậy sinh viên nên nộp bài càng sớm càng tốt. Sinh viên sẽ tự chịu rủi ro nếu nộp bài sát hạn chót. Khi quá thời hạn nộp bài, hệ thống sẽ đóng nên sinh viên sẽ không thể nộp nữa. Bài nộp qua email không được chấp nhận.

5 Harmony

Trong đề thi có thể có các câu hỏi liên quan đến nội dung bài tập lớn (phần mô tả câu hỏi sẽ được nêu rõ là dùng để harmony cho bài tập lớn, nếu có) . Điểm của các câu hỏi harmony sẽ được scale về thang 10 và sẽ được dùng để tính lại điểm của các bài tập lớn. Cụ thể:

- Gọi x là điểm bài tập lớn.
- Gọi y là điểm của các câu hỏi harmony sau khi scale về thang 10.

Điểm cuối cùng của bài tập lớn sẽ được tính theo công thức trung bình điều hòa sau:

$$\mathbf{Assignment\_Score = 2 * x * y / (x + y)}$$

6 Gian lận

Bài làm của sinh viên sẽ được kiểm tra sự giống nhau thông qua hệ thống phát hiện gian lận MOSS. Tất cả các bài làm có **tỷ lệ giống nhau lớn hơn 30%** đều được xem là gian lận.