Bài tập lớn 2
Hiện thực Cache

TS. Nguyễn Hứa Phùng

Tháng 3/ 2021

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 cây AVL vào vấn đề tìm kiếm trong một cache đơn giản

2 Giới thiệu

Trong một máy tính thực, tốc độ truy xuất của bộ nhớ chính rất thấp so với tốc độ bộ vi xử lý của nó. Để cải thiện hiệu suất hoạt động, bộ nhớ cache 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 Cơ chế thay thế cache

Vì kích thước của cache là có giới hạn, khi cache đầy và một dữ liệu mới được thêm vào cache nhằm phục vụ các yêu cầu trong tương lai, một dữ liệu hiện có trong cache được lựa chọn để xóa đi nhằm nhường chỗ lại cho dữ liệu mới. Việc lựa chọn này được thực hiện bởi một cơ chế thay thế cache được liệt kê dưới đây:

- 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
- LRU (Least Recently Used): dữ liệu được sử dụng trễ nhất sẽ được chọn
- MSU (Most Recently Used): dữ liệu được sử dụng gần đây nhất sẽ được chọn
- LFU (Least Frequently Used): dữ liệu ít được sử dụng nhất sẽ được chọn
- RR (Random Replacement): một dữ liệu ngẫu nhiên 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 (đọc/ghi) dữ liệu trong bộ nhớ chính.

3.1.1 Các phương thức của cache

Để hoàn thành bài tập lớn này, các em cần phải hiện thực các phương thức sau của lớp Cache:

- **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** and **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 **cont** cho dữ liệu đang liên kết với **addr**. Ngược lại, nếu không tìm thấy, phương thức này đưa bộ **addr** và **cont** vào trong cache. Phương thức này trả về phần tử bị đưa ra khỏi cache, nếu có, ngược lại trả về NULLL. Lưu ý rằng **cont** cùng với địa chỉ **addr** của nó được đưa vào cache, 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 giảm dần của thời gian sống của mỗi phần tử trong cache. 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 thứ tự preorder của cây AVL. Đây là cây được sử dụng để dễ dàng tìm kiếm một địa chỉ trong cache.

- **void inOrder()**: in giá trị của các phần tử trong cache theo thứ tự inorder của cây AVL

Khi một phần tử mới được đưa vào **full** cache (thông qua phương thức **put** hoặc **write**), chính sách thay thế **FIFO** 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 AVL 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 tương ứng địa chỉ **addr**, nếu có trong cache, 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 (print, preorder, inorder) được sử dụng chỉ để kiểm tra cách hiện thực của các em.

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 để biểu diễn 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 Trình tự thực hiện

Để hoàn thành bài tập lớn này, các em 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. Các em 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. Nếu vi phạm yêu cầu này, bài của các em sẽ không được chấm.

4 Nộp bài

Các em chỉ nộp 2 files: Cache.h và Cache.cpp, trước thời hạn được đưa ra trong đường dẫn "Assignment 2 Submission". Có một số testcase đơn giản được sử dụng để kiểm tra bài làm của các em nhằm đảm bảo rằng kết quả của em có thể biên dịch và chạy được. Các em 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 em nộp bài cùng một lúc, vì vậy các em nên nộp bài càng sớm càng tốt. Các em sẽ phải tự chịu rủi ro nếu nộp bài sát hạn chót. Vi Khi quá thời hạn nộp bài, hệ thống sẽ đóng nên các em sẽ không thể nộp nữa. Bài nộp qua email sẽ không được chấp nhận.

5 Xử lý gian lận

Các em phải tự mình hoàn thành bài tập lớn này này và phải ngăn không cho người khác đánh cắp kết quả của mình. Nếu không, các em sẽ bị xử lý theo quy định của trường vì gian lận.