HỆ ĐIỀU HÀNH

GV LT: Thái Hùng Văn

BÁO CÁO CUỐI KỲ

Nhóm

19120644 Họ tên: Lê Đức Tâm

19120622 Họ tên: Nguyễn Minh Phụng

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

MỤC LỤC

CÂU 1

ĐỀ BÀI

Trình bày kỹ thuật phân trang và bộ nhớ ảo. Khảo sát và cho biết giá trị của các thông số quản lý (số bit quản lý địa chỉ ô nhớ, kích thước trang, số bit tối thiểu để quản lý các offset trong trang, số khung trang vật lý, số khung trang logic tối đa trên không gian tiến trình, kích thước phần bộ nhớ ảo,…) trên một hệ thống máy tính cụ thể (và diễn giải)

BÀI LÀM

Phân trang là một kĩ thuật quản lý bộ nhớ giúp loại bỏ nhu cầu cấp phát liên tục của bộ nhớ vật lý (cấp phát không liên tục). Lược đồ này cho phép không gian địa chỉ vật lý của một tiến trình không liền nhau.

- Địa chỉ logic hoặc địa chỉ ảo (bit): Một địa chỉ do CPU tạo ra
- Không gian địa chỉ logic hoặc không gian địa chỉ ảo (word hoặc byte): Tập hợp tất cả các địa chỉ logic được tạo bởi một chương trình
- Địa chỉ vật lý (bit): Một địa chỉ thực sự có sẵn trên đơn vị bộ nhớ
- Không gian địa chỉ vật lý (word hoặc byte): Tập hợp tất cả các địa chỉ vật lý tương ứng với các địa chỉ logic

Việc ánh xạ từ địa chỉ ảo sang địa chỉ vật lý được thực hiện bởi đơn vị quản lý bộ nhớ (Memory Management Unit - MMU) là một thiết bị phần cứng và sự liên kết này được gọi là kỹ thuật phân trang.

- Không gian địa chỉ vật lý được chia thành các khối có kích thước cố định, được gọi là khung (frame).
- Không gian địa chỉ logic cũng được chia thành các khối có kích thước cố định, được gọi là các trang (page).
- Kích thước page và frame bằng nhau

Địa chỉ do CPU tạo ra được chia thành:

- Số trang – page number (p): Số bit đại diện cho các trang trong không gian địa chỉ logic
- Offset trang – page offset (d): Số bit đại diện cho vị trí offset của page trong không gian địa chỉ logic

Địa chỉ thực được chia thành

- Số khung – frame number (f): Số bit đại diện cho các khung của Không gian địa chỉ vật lý
- Offset khung – frame offset (d): Số bit biểu thị vị trí offset của frame trong không gian địa chỉ vật lý

Một cấu trúc dữ liệu được gọi là bảng trang (page table) được sử dụng để theo dõi mối quan hệ giữa một trang của quy trình với một khung trong bộ nhớ vật lý.

Việc triển khai bảng trang (page table) có thể được thực hiện bằng cách sử dụng các thanh ghi chuyên dụng. Nhưng việc sử dụng đăng ký cho bảng trang chỉ thỏa mãn nếu bảng trang nhỏ. Nếu bảng trang chứa một số lượng lớn các entry thì chúng ta có thể sử dụng TLB (translation Look-aside buffer) một bộ đệm phần cứng đặc biệt, nhỏ gọn, tra cứu nhanh.

Mỗi mục nhập trong TLB bao gồm hai phần: thẻ và giá trị.
Khi bộ nhớ này được sử dụng, một mục sẽ được so sánh với tất cả các thẻ đồng thời. Khi mục tương tự được tìm thấy thì giá trị tương ứng sẽ được trả về.
Khi một quy trình được thực thi, các page tương ứng của nó sẽ được tải vào bất kỳ frame bộ nhớ khả dụng nào. Khi máy tính hết RAM, hệ điều hành sẽ di chuyển các page bộ nhớ thích hợp hoặc không mong muốn sang bộ nhớ phụ để giải phóng RAM cho các tiến trình khác và đưa chúng trở lại khi chương trình cần.

Ưu và nhược điểm:

- Phân trang làm giảm phân mảnh ngoại vi, nhưng vẫn bị phân mảnh nội vi.
- Phân trang đơn giản, dễ thực hiện và được coi là một kỹ thuật quản lý bộ nhớ hiệu quả.

- Do kích thước của các trang và khung bằng nhau, việc hoán đổi trở nên rất dễ dàng.
- Bảng trang yêu cầu thêm dung lượng bộ nhớ, do đó có thể không tốt cho hệ thống có RAM nhỏ.

Bộ nhớ ảo

Bộ nhớ ảo là kĩ thuật mở rộng vùng nhớ, giúp cho người dùng được giải phóng hoàn toàn khỏi mối bận tâm về giới hạn bộ nhớ. Trích xuất một phần của bộ nhớ phụ, vùng nhớ đó sẽ được xem như là bộ nhớ ảo hỗ trợ cho các thao tác của RAM, giả lập RAM.

Nó là một kỹ thuật được thực hiện bằng cả phần cứng và phần mềm. Cho phép xử lý một tiến trình không được nạp toàn bộ vào bộ nhớ vật lý. Bộ nhớ ảo mô hình hoá bộ nhớ như một bảng lưu trữ rất lớn và đồng nhất, tách biệt hẳn khái niệm không gian địa chỉ và không gian vật lý. Người sử dụng chỉ nhìn thấy và làm việc trong không gian địa chỉ ảo, việc chuyển đổi sang không gian vật lý do hệ điều hành thực hiện với sự trợ giúp của các cơ chế phần cứng cụ thể.

Bộ nhớ ảo được triển khai bằng cách sử dụng phân trang theo nhu cầu (demand paging) hoặc Phân đoạn theo nhu cầu (demand segmentation).

Một hệ thống phân trang theo yêu cầu khá giống với hệ thống sử dụng kỹ thuật phân trang kết hợp với kỹ thuật swapping. Một tiến trình được xem như một tập các trang, thường trú trên
bộ nhớ phụ (bộ nhớ ảo – thường là đĩa cứng). Khi cần xử lý, tiến trình sẽ được nạp vào bộ nhớ chính nhưng chỉ đối với những trang cần thiết ở thời điểm run-time. Như vậy một trang chỉ được nạp vào bộ nhớ chính khi có yêu cầu.

Trong khi thực thi một chương trình, nếu chương trình tham chiếu đến một trang không có sẵn trong bộ nhớ chính vì nó đã bị hoán đổi gần đây, bộ xử lý sẽ coi tham chiếu bộ nhớ không hợp lệ này là lỗi trang (page fault) và chuyển quyền điều khiển từ chương trình sang hệ điều hành yêu cầu trang trở lại bộ nhớ.

Ưu và nhược điểm:

- Bộ nhớ ảo lớn.
- Sử dụng bộ nhớ hiệu quả hơn.
- Không có giới hạn về mức độ đa chương trình.
- Số lượng bảng và chi phí xử lý các ngắt trang lớn hơn so với trường hợp của các kỹ thuật quản lý phân trang đơn giản.

Thuật toán thay thế trang

Thuật toán thay thế trang được áp dụng khi gặp lỗi trang, xem xét thông tin hạn chế đồng thời cố gắng chọn trang nào nên được thay thế để giảm thiểu tổng số trang bị bỏ lỡ, đồng thời cân bằng nó với chi phí lưu trữ chính và thời gian xử lý của chính thuật toán. Có nhiều thuật toán thay thế trang khác nhau. Chỉ tiêu đánh giá thuật toán thông qua khả năng chạy trên một chuỗi các địa chỉ cần truy xuất và giảm thiểu số trang lỗi (page fault) phát sinh.

Thuật toán First In First Out (FIFO)

- Trang cũ nhất trong bộ nhớ chính là trang sẽ được chọn để thay thế.
- Dễ dàng thực hiện, thay thế các trang từ phần đuôi và thêm các trang mới ở phần đầu.

Thuật toán Optimal page

- Thuật toán thay thế trang tối ưu có tỷ lệ lỗi trang thấp nhất trong tất cả các thuật toán. Các thuật toán thay thế trang tối ưu OPT, MIN.
- Thay thế trang có khả năng không được sử dụng trong thời gian dài nhất.

Thuật toán Least Recently Used (LRU)

- Trang không được sử dụng trong thời gian dài nhất trong bộ nhớ chính là trang sẽ được chọn để thay thế.
- Dễ dàng thực hiện, thay thế các trang bằng cách nhìn lại mốc thời gian.

Thuật toán Page Buffering

- Để quá trình bắt đầu nhanh chóng, hãy giữ một nhóm các khung trống.
- Trên trang bị lỗi, hãy chọn một trang để thay thế.

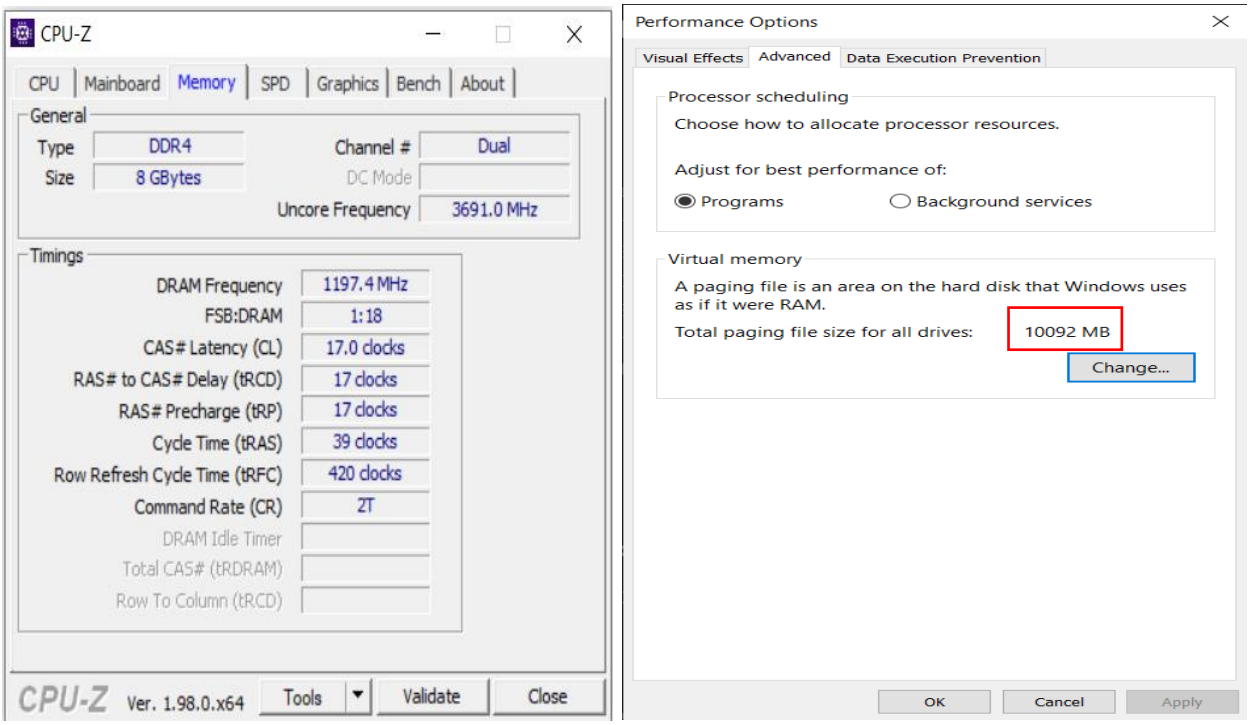
- Viết trang mới trong khung của nhóm khung trống, đánh dấu bảng trang và khởi động lại quá trình.
- Bây giờ ghi trang tương ứng ra khỏi đĩa và đặt khung chứa trang đã thay thế vào vùng trống.

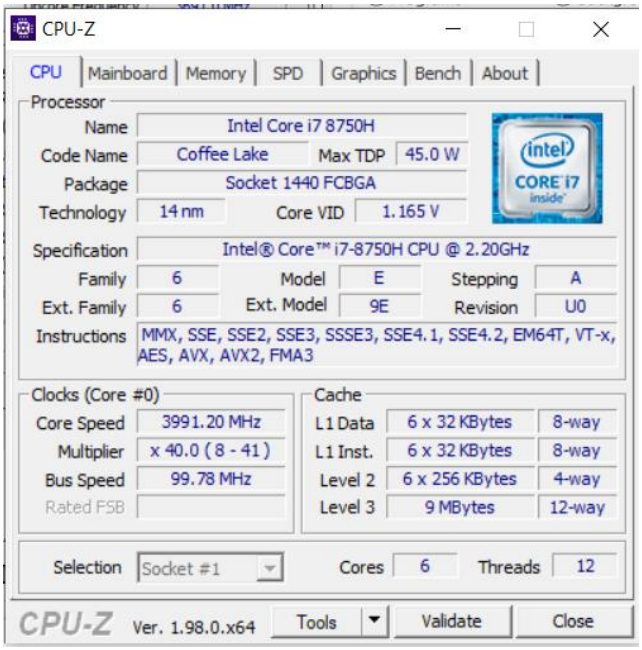
Thuật toán Least frequently Used (LFU)

- Trang có số lượng nhỏ nhất là trang sẽ được chọn để thay thế.
- Thuật toán này gặp phải tình huống trong đó một trang được sử dụng nhiều trong giai đoạn đầu của quy trình, nhưng sau đó không bao giờ được sử dụng lại.

Thuật toán Most frequently Used (MFU): Thuật toán này dựa trên lập luận rằng trang có số lượng nhỏ nhất mới có thể được đưa vào tuy nhiên vẫn chưa được sử dụng.

Khảo sát:





Ta có:

Kích thước trang = kích thước khung trang = $4\ KB$

Không gian vật lý (Main memory – RAM) = $8\ GB$

Không gian logic (Kích thước bộ nhớ ảo) = $10092\ MB$

Ta có không gian vật lý $= 8\ GB = 2^{33}\ B \Rightarrow$ số bit quản lý địa chỉ ô nhớ $= 33$ bit

Ta có kích thước trang $= 4\ KB = 2^{12}\ B \Rightarrow$ số bit tối thiểu để quản lý các offset trong trang $=\ 12$ bit

Số khung trang vật lý $= \frac{\text{không gian vật lý}}{\text{Kích thước khung trang}} = \frac{8\ GB}{4\ KB} = \frac{2^{33}\ B}{2^{12}\ B} = 2^{21} =$ 2097152 khung trang

Số trang logic tối đa trên không gian tiến trình $= \frac{\text{Không gian logic}}{\text{Kích thước trang}} = \frac{10092\ MB}{4\ KB} =$ $\frac{3 \times 29^2 \times 2^{22}\ B}{2^{12}\ B} = 3 \times 29^2 \times 2^{10} = 2583552$ trang

CÂU 2

ĐỀ BÀI

Xây dựng hệ thống chương trình cho phép liên lạc giữa các tiến trình trong 1 máy và /hoặc trên các máy *(với các hệ điều hành có thể khác nhau)*; sao cho chúng có khả năng phòng ***Deadlock*** hoặc tránh cùng rơi vào ***Critical section***

MÔ TẢ

Hệ thống giám sát /can thiệp thời gian sử dụng máy tính của trẻ em bao gồm 2 chương trình chính:

Chương trình C (for Children): chạy trên máy laptop /desktop hệ điều hành Windows của trẻ và được đặt ở chế độ Autorun (tự động chạy khi bật máy), thực hiện các việc:

1. Lấy mật khẩu (từ bàn phím)
2. Nếu chuỗi nhập là mật khẩu của phụ huynh: Chương trình đợi đến 7 giây sau thì mới quay lại hỏi lại mật khẩu (thực hiện lại bước 1) // lúc này là phụ huynh sử dụng máy chứ không phải trẻ!
3. Nếu không phải là mật khẩu của phụ huynh:
   3.1. Kiểm tra xem thời điểm hiện tại có nằm trong khung thời gian trẻ chưa được dùng máy hay không:
      3.1.1. *Nếu đang trong khoảng thời gian trẻ chưa được dùng máy:*
         Thông báo tới khi nào mới được dùng (hiển thị ra màn hình và /hoặc nói ra loa), sau đó thực hiện song song 2 việc: (1): kiểm tra xem đã đủ 15 giây chưa kể từ lúc thông báo xong & nếu đã đủ thì chương trình tự tắt máy (shutdown hệ điều hành - không cho người dùng can thiệp) – (2) thực hiện lại từ đầu việc 1 & 2 (tức nếu người dùng kịp nhập đúng mật khẩu phụ huynh thì không tắt máy mà thực hiện 2 - đợi đến 7 giây sau…)
      3.1.2. Ngược lại (đang trong khoảng thời gian trẻ được dùng máy):

3.1.2.1. *Nếu mật khẩu không phải mật khẩu của trẻ:* thực hiện lại việc hỏi và kiểm tra mật khẩu (thực hiện lại C0) cho đến lần nhập sai mật khẩu thứ 3 thì đặt thời gian không được dùng máy là 10 phút kể từ thời điểm hiện tại rồi tắt máy.

3.1.2.2. *Ngược lại (đúng mật khẩu của trẻ):* Đọc thông tin về khung giờ được dùng (a) và Thông báo còn bao nhiêu phút nữa máy sẽ, sau đó chạy ở chế độ giám sát thực hiện cùng lúc các việc: (1) Sau mỗi 0.13 giây lưu lại nội dung các phím đã gõ, (2) thực hiện (a) và thấy có thay đổi (do cha /mẹ chạy tiến trình P và điều chỉnh) thì cập nhật thông tin và thông báo còn mấy phút nữa tắt máy, (3) kiểm tra thấy còn 1 phút đến thời điểm tắt máy thì thông báo và còn 0 phút thì tắt máy. Thông tin về khung giờ sử dụng sẽ được cập nhật

Thông tin sẽ được lưu trên cloud Firebase sẽ có hai loại là **Schedule** cho phần lịch cho phép dùng máy và **History** cho phần lịch sử sử dụng máy của trẻ.

Trong Schedule: mỗi lịch sẽ có các trường: date dùng để thể hiện ngày, duration dùng để thể hiện thời gian tối đa 1 lần máy có thể được mở(tính theo phút), interruptTime thể hiện thời gian ngắt(tính theo phút), sum thể hiện thời gian tổng trong 1 lần mở máy, timeStart thể hiện thời gian bắt đầu có thể mở máy (HH:mm), timeEnd thể hiện thời gian kết thúc có thể mở máy(HH:mm)

Ví dụ cụ thể, với nội dung **Schedule** như sau: date là “12/01/2022”, TimeStart là 14:20 đến timeEnd là 17:20, duration bằng 60, interruptTime bằng 12 sum bằng 126 thì các khung giờ được dùng là: Trong khoảng thời gian từ 07:30 đến 11:30 ngày “12/01/2022” có thể sử dụng máy, nhưng mỗi lần bật máy thì chỉ được dùng tối đa 60 phút – sau đó máy sẽ không hoạt động cho đến khi đã ngắt đủ 12 phút, đồng thời khi đã dùng đủ 126 phút thì máy cũng sẽ không chịu chạy nữa.

Trong History: Mỗi lịch sử sẽ có các trường: date để thể hiện ngày, keyLog để thể hiện các phím được nhấn trong lần sử dụng đó,

timeStart để thể hiện thời gian mở máy, timeEnd để thể hiện thời gian tắt máy.

Ví dụ cụ thể, với nội dung **History** như sau: date là "12/01/2022", TimeStart là 14:20 đến timeEnd là 17:20, keyLog lưu chuỗi ký tự mà trẻ nhập trong khoảng thời gian đó: Trong khoảng thời gian từ 14:20 đến 17:20 ngày 12/01/2022 trẻ đã đăng nhập máy và chuỗi các ký tự mà trẻ đã nhập là chuỗi keyLog, nếu chuỗi keyLog và TimeEnd đang là -1 thì trẻ đang sử dụng máy ngay lúc đó.

Chương trình P (for Parent): Chạy trên điện thoại hệ điều hành Android, thực hiện các việc:

1. Lấy mật khẩu (từ bàn phím)
2. *Nếu chuỗi nhập không phải là mật khẩu của phụ huynh:* Chương trình xuất ra thông báo sai mật khẩu rồi cho phép nhập lại
3. *Nếu chuỗi nhập là mật khẩu của phụ huynh:* Vào giao diện chính của chương trình, những lần sau chỉ cần mở ứng dụng không cần đăng nhập lại, nếu muốn đăng nhập lại phải đăng xuất chương trình
   - 3.1. *Trong giao diện chính của chương trình có 2 tab lịch sử thể hiện lịch sử sử dụng máy và lịch thể hiện các mốc thời gian có thể sử dụng máy:*
     - 3.1.1. *Tab lịch:*
       - 3.1.1.1. Hỏi người dùng ngày muốn xem lịch sau khi nhập ngày thì có thể thêm lịch, xóa lịch, sửa lịch, xem chi tiết lịch. Từng lịch thể hiện ngày, thời gian bắt đầu và kết thúc
     - 3.1.2. *Tab lịch sử:*
       - 3.1.2.1. Hỏi người dùng ngày muốn xem lịch sử sau khi nhập ngày thì cho phép người dùng xem lịch sử mà trẻ mở máy vào các ngày (ngày mở, thời gian bắt đầu, thời gian kết thúc). Từng lịch sử có thể cho người dùng thông tin bàn phím nhập trong lúc đó. Nếu lịch sử có giờ kết thúc bằng -1 thì tức là trẻ đang online).

Lưu ý: Do vẫn chưa có thể can thiệp vào hệ thống để ngăn các chức năng xóa process từ phía người dùng nên nếu người dùng nhấn phím tắt Alt+F4 thì chương trình sẽ tự động bị hủy.

CÁCH GIẢI QUYẾT DEADLOCK VÀ CRITICAL SECTION (MÃ GIẢ)

- **Cách giải quyết:**
  - Sử dụng cờ hiệu
- **Mã giả (Đoạn code bị miền găng)**

```
if(Các dữ liệu hợp lệ)
    if(trường hợp thêm lịch mới){
        Schedule schedule_final → new Schedule(dữ liệu nhập và cho biến cờ
hiệu ban đầu là "0");
           //add dữ liệu vào Cloud
        scheduleDBM.add(schedule_final).addOnSuccessListener(suc ->
        {
            Thêm thành công thì hiện thông báo
        }).addOnFailureListener(er ->
        {
            Thêm thất bại thì hiện lỗi
        });
    }
    else {//trường hợp chỉnh sửa lịch
        HashMap<String,Object> hashMap→new HashMap<>();
        hashMap → dữ liệu chỉnh sửa mới+ biến cờ hiệu là "1";
        scheduleDBM.getFlag(schedule.getKey()).addOnSuccessListener(new
OnSuccessListener<DataSnapshot>() {//thành công xin cờ hiệu ở lịch mà mình
muốn sửa
            @Override
            public void onSuccess(DataSnapshot dataSnapshot) {
                if (cờ hiệu lấy được == "0") {
                    db.update(hashMap)// update dữ liệu cũ bằng dữ liệu
mới
                      Hiện thông báo "Cập nhật thành công"

                } else {//cờ hiệu lấy được =="1"
                    Hiện thông báo "Dữ liệu đang được truy xuất"
                }
            }
        }).addOnFailureListener(new OnFailureListener() {//Thất bại khi
xin cờ hiệu mà mình muốn sửa
            @Override
            public void onFailure(@NonNull Exception e) {
                Hiện thông báo lỗi
            }
        });
    }
```

```
    Quay trở lại màn hình Lịch
}
```

- **Ưu điểm:**
  - Code dễ hiểu, dễ dàng chỉnh sửa và phát triển
  - Giải quyết được tình trạng deadlock
- **Khuyết điểm:**
  - Không đảm bảo không có 2 tiến trình nào ở cùng miền găng cùng lúc (mutual exclusion) do code kiểm tra cờ hiệu và dành quyền cũng nằm trong miền găng
- **Hướng phát triển thuật toán:**
  - Sử dụng method runTransaction(…) được hỗ trợ bởi Firebase để chịu trách nhiệm lắng nghe dữ liệu và trả về Transaction.Result nếu dữ liệu mới tại vị trí muốn thay đổi nên được tiến hành hay hủy bỏ.

VIDEO DEMO

- Link youtube:

- Link drive (dự phòng):

TÀI LIỆU THAM KHẢO

CÂU 1:

- Giáo trình Hệ điều hành
- Operating System - Memory Management (tutorialspoint.com)
- Operating System - Virtual Memory (tutorialspoint.com)
- Paging in Operating System - GeeksforGeeks
- Virtual Memory in Operating System - GeeksforGeeks

- Memory Hierarchy Design - Part 6. The Intel Core i7, fallacies, and pitfalls - EDN

CÂU 2:

- https://stackoverflow.com/questions/60775018/getting-data-from-firebase-in-a-synchronized-way
- https://stackoverflow.com/questions/47847694/how-to-return-datasnapshot-value-as-a-result-of-a-method/47853774
- https://firebase.google.com/docs/reference/android/com/google/firebase/database/Transaction.Handler