Sprawozdanie

Algorytmy zastępowania stron

Krzysztof Mola 264055

13.12.2022

Spis treści

Informacje ogólne dotyczące projektu:

Pamięć wirtualna jest oddzieleniem pamięci fizycznej od pamięci logicznej dostępnej dla użytkownika. U podstaw tej idei leży obserwacja, że w każdej chwili tylko część pamięci procesu musi znajdować się w pamięci operacyjnej komputera - ta część, która akurat jest używana przez proces. Zadaniem pamięci wirtualnej jest imitowanie pamięci, w celu uniknięcia wykorzystania nadmiarowej ilości zasobów.

Celem poniższego projektu jest zapoznanie się z algorytmami zastępowania stron stosowanymi do wybierania zasobów, które będą zamieniane w trakcie wykonywania procesu, omówienie symulacji wybranych algorytmów i ich działania dla zamkniętej puli zadań. Językiem wykorzystanym do przeprowadzenia symulacji będzie Python - wysoko poziomowy, dynamiczny język ogólnego przeznaczenia. Wybrany on został ze względu na czytelną składnię i prostotę implementacji algorytmów zastępowania stron.

Algorytmy wykorzystane przy realizacji projektu:

Algorytmy zastępowania stron zajmują się właśnie znalezieniem „ofiary” na podstawie dostępnych wytycznych. Wybór odpowiedniego algorytmu może mieć znaczący wpływ na efektywność pamięci wirtualnej. Przy realizacji projektu wykorzystane zostały dwa algorytmy zastępowania stron:

Algorytm FIFO (First In, First Out):

W tym algorytmie usuwana z pamięci będzie strona, która przebywa w niej przez najdłuższy okres czasu. Strony znajdujące się w pamięci fizycznej tworzą coś na kształt kolejki, a „ofiary” – strony do usunięcia brane są z jej początku. Algorytm ten nie bierze pod uwagę właściwości obsługiwanych procesów. Jego największą zaletą jest prostota jego implementacji.

W projekcie jest on wykorzystany na zasadzie procesu, w którym dla poszczególnych stron i ilości ramek, przy szukaniu strony, którą trzeba będzie usunąć w danym kroku będziemy sprawdzali, która z ramek zawierających aktualnie daną stronę przez najdłuższy czas znajduje się w pamięci. Po wyrzuceniu jej zostanie ona zastąpiona nową stroną, dodaną na koniec kolejki, a reszta stron przesunie się o jedno miejsce do przodu.

Algorytm LRU (Least Recently Used):

W tym algorytmie usuwana z pamięci będzie strona, która nie była wykorzystywana przez najdłuższy okres czasu. Bierze ona pod uwagę przeszłe stany. Jej implementacja jest bardziej skomplikowana od poprzednika, ze względu na to, że każde odwołanie musi zostawiać po sobie jakiś ślad na podstawie, którego ocenimy jak długo faktycznie przebywa ona w pamięci.

W projekcie jest on wykorzystany na zasadzie procesu, w którym dla poszczególnych stron i ilości ramek, przy szukaniu strony, którą trzeba będzie usunąć w danym kroku będziemy sprawdzali, która z ramek zawierających aktualnie daną stronę przez najdłuższy czas nie została w żaden sposób wykorzystana. Po wyrzuceniu jej zostanie ona zastąpiona nową stroną, dodaną na koniec kolejki, a dla reszty stron w dalszym ciągu będzie zapamiętywana ich żywotność, aby kiedy znowu trzeba będzie zamienić stronę znana była informacja, która z pozostałych jest aktualnie najdłużej nieużywaną.

Opis przeprowadzenia symulacji:

Program wykorzystany do przeprowadzenia symulacji napisany jest w języku Python, w sposób obiektowy. Oznacza to, że projekt składa się ze współpracujących ze sobą podprogramów połączonych ze sobą w pliku głównym.

Plik główny (main.py):

Odpowiedzialny jest za połączenie ze sobą podprogramów niezbędnych do wykonania symulacji. Odpowiedzialny jest on za wykonanie skryptów zliczających ilość brakujących stron dla zadanych danych i przeprowadzenie stosownych obliczeń niezbędnych w późniejszym etapie do porównania działania obydwu zaimplementowanych algorytmów. Zapisuje on również wyniki przeprowadzonych operacji osobno dla obydwu algorytmów.

Plik generujący przykładowe dane (stworz_dane.py):

Algorytm ten tworzy 50 plików o nazwach (1-50) z rozszerzeniem .txt i zapisuje je w folderze **dane.** Każdy z plików zostanie uzupełniony stoma liczbami z zakresu 1-15 z wykorzystaniem biblioteki random, zapisanymi jedna pod drugą. Liczby te będą symulowały strony i będą wykorzystywane do zliczania stanów, w których w ramce podczas trwania procesu będzie brakowało danej strony.

Plik obliczający brakujące strony FIFO (algorytm_FIFO_podmiana_stron.py):

Służy on do obliczania ilości brakujących stron dla danego pliku z danymi wygenerowanymi przez **stworz_dane.py.** Ilości brakujących stron zliczane są osobno dla różnych ilości dostępnych ramek (3, 5, 7) i zapisywane do poszczególnych plików znajdujących się w folderze **wyniki_algorytm_FIFO**. Nazwy plików przechowujących wyniki zadanego algorytmu mają następujące nazwy: **3.txt, 5.txt, 7.txt.** Wyniki te zostaną wykorzystane do przeprowadzenia obliczeń statystycznych potrzebnych do porównania jakości obydwu algorytmów.

Plik obliczający brakujące strony LRU (algorytm_LRU_podmiana_stron.py):

Służy on do obliczania ilości brakujących stron dla danego pliku z danymi wygenerowanymi przez **stworz_dane.py.** Ilości brakujących stron zliczane są osobno dla różnych ilości dostępnych ramek (3, 5, 7) i zapisywane do poszczególnych plików znajdujących się w folderze **wyniki_algorytm_LRU**. Nazwy plików przechowujących wyniki zadanego algorytmu mają następujące nazwy: **3.txt, 5.txt, 7.txt.** Wyniki te zostaną wykorzystane do przeprowadzenia obliczeń statystycznych potrzebnych do porównania jakości obydwu algorytmów.

Plik wykonujący obliczenia statystyczne (kalkulacje.py):

Jego zadaniem jest podsumowanie zebranych danych w postaci policzenia średniej brakujących stron dla poszczególnych ilości dostępnych ramek (3, 5, 7). Liczy on również odchylenie standardowe dla poszczególnych ilości dostępnych ramek (3, 5, 7). Wyniki jego operacji zostają zapisane do pliku **obliczenia.txt** skąd bezpośrednio można już je odczytać**.**

Porównanie algorytmów:

Algorytmy te będą porównywane z uwzględnieniem dwóch kryteriów:

- Średniej ilości brakujących stron dla poszczególnych ilości dostępnych ramek

- Odchylenia standardowego dla powyższych średnich

Lepszym będzie algorytm, którego:

- Średnia ilości brakujących stron będzie mniejsza – oznacza to, że dany algorytm będzie tym lepszy im mniejsza będzie ilość błędów.

- Odchylenie standardowe od średniej - Im mniejsza wartość odchylenia tym obserwacje są bardziej skupione wokół średniej. Im mniejsze odchylenie standardowe tym lepszy algorytm.

Poniżej zamieszczone zostaną wyniki dla poszczególnych algorytmów będących wynikiem wykonania programu:

Ilość dostępnych ramek	Średnia brakujących stron FIFO z odch std	Średnia brakujących stron LRU z odch std
3	40,23 +- 4,09	40,26 +- 4,51
5	34,11 +- 4,97	34,11 +- 4,74
7	28,02 +- 4,65	27,9 +- 4,41

Rysunek 1: Porównanie średnich ilości błędów obydwu algorytmów z uwzględnieniem odchyleń standardowych

Analiza i interpretacja wyników:

Z „pomiarów” przeprowadzonych za pomocą programy symulacyjnego dla wygenerowanych danych wynik jest następujący:

- Niezależnie od ilości dostępnych ramek algorytmy są bardzo zbliżone do siebie względem średnich ilości brakujących stron. Oznacza to, że względem średnich algorytmy są praktycznie równe względem siebie w kwestii optymalności.

- Biorąc pod uwagę analizę odchyleń standardowych od średnich wynik jest bardziej korzystny dla algorytmu LRU, aczkolwiek w tym wypadku różnica też jest praktycznie nieznaczna.

Analizując wyniki możemy dojść do wniosków, że algorytmy są bardzo podobne w kwestii optymalności. Wyniki te jednak mogłyby różnić się dla bardziej specyficznych danych – każdy algorytm posiada jakąś swoją mocną stronę. Podając losowe dane wyniki te można jednak uznać za w pewnym stopniu wyważone.

Za ostateczny wniosek można uznać fakt, że algorytmy te są bardzo zbliżone do siebie w kwestii optymalności.