Sprawozdanie

Systemy Operacyjne K00-64a

Ptashkohrai Andrii 265535

20.01.2023

Algorytmy przydziłu czasu procesora

- FCFS

-SJF

Algorytmy zastępowania stron

-FIFO

-LRU

Cel projektu

Celem tego projektu jest zapoznanie z algorytmami przedziłu czasu procesora FCFS(First Come First Serve) i SJF(Shortest Job First), również zapoznanie z algorytmami zastępowania stron FIFO(First In First Out) i LRU(Least Recently Used), zasadami ich dziłania oraz mechanizmami ich implementacji. Muszę też przetestować wszystkie algorytmy i w przpadku algorytmów przedłu czasu procesora prównać według dwóch cech: średni czas wykonania procesów i średni czas oczekiwania, w przypadku algorytmów zastępowania stron muszę ich porównać według średniej ilości powtarzalnych liczb i ilości nipowtarzalnych liczb.

Opis procedury testowania algorytmów

W celu przetestowania tych algorytmów napisałem w Pythonie program który składający się z 6 modułów mianowicie składa się z generatora, który generuje 50 procesów z czasem pzybycza i czasem realizacjii dla pierwszych dwoch algorytmów i zapisuje to w plik test\_data\_1.txt. Wówczas ten generatoe generuje 50 liczb(stron) dla dwoch innych algorytmów i sapisuje to w plik test\_data\_2.txt.

Moduł main.py jest modułem głuwnym. Rónież ten program mieście się realizacje algorytmów, oni zapisani w plikach FCFS.py SJF.py FIFO.py LRU.py odpowiednio.

Moduły te zwracają nam średni czas wykonywania procesów, średni czas ociekiwania dla 50 procesów, średną ilość powtarzalnych i nie powtarzalnych liczb. Nastęmpnie my wyświtlamy te wyniki w konsoli.

Moduły samych algorytmów dziłają w ten sposób:

**FCFS**: otrzymuje nazwę plika i parsuję procesy na listę list, która ma trzy zminne – nazwę procesa, czas jego przybycza i czas wykonania. Następnie sortujemy tę listę według czasów przybycza i krok po kroku iterujemy przez każdy algorytm, dadając czas ich wykonania i obliczając dla każdego czas wykonania procesa i czas oczekiwania .Następnie dzilimy tę informację przez ilość naszych procesów i uzyskujemy odpowiedzi.

**SJF**: dziła w podobny sposób, ale jest bardziej skomplikowany. Nie możemy po prostu posortować nasze procesy według szybkości ich wykonania. Po kożkedj wykonanej iteracjii musimy posortować wszystkie procesy według ich czasu pracy i dopiero wtedy wykonać najkrótrzy z nich. Odbywa się to po to abyśmy nie urochomiali algorytmu który jeszcze nie trafił do kolejki co będzie dużym błędem. Odbywa to za pomocy specjalnej funkcjii ktora otrzymuje całą listę procesów oraz aktualny czas. Ta funkcja zwraca posortowaną listę procesów których czas przebycia jest mnijszy lub równy czasowi bieżącemu. Następnie znadujemy ten proces na prawdziwej liście procesów i umieściamy go na początek, dodajemy czas jego wykonania i usuwamy go z listy, ponieważ nie będziemy już go potrzebować.

Dodatkowo w tych dwoch algorytmach jest zapiezpieczenie przed sytuacją, gdy w kolejce nie ma algorytmów i jesteśmy zmuszeni czkać na przybycie kolejnego.W takiej sytuacji zrównujemy nasz aktualny czas z czasem nadejścia kolejnego procesu i kontynujemy nasz algorytm.

**FIFO**: jak i w przypadku FCFS otważamy nasz plik i parsujemy nasze liczby na listę. Natepnie wypełniamy nasze roboczą listę(w tej liście dodajemy albo zmieniamy liczby) i plusujemy 1 do ogólnej liczby missów. Dalej my sprawdzamy czy następna liczba w liście z liczbamy, jest podobna do liczb w roboczej liście, jeśli mamy podobne liczby to plusujemy 1 do ogólnej liczy hitów. Następnie jeśli my nie mamy podobnych liczb w naszej liście roboczej, to zaminiamy otatnio dodanu liczbu na aktualnu liczbu i plusujemy 1 do ogólnej liczy missów. Na koniec wyznaczamy hitrate i missrate dzieląc ilość hitów i missów na długość naszej listy z liczbami.

**LRU**: dziła w podobny sposób jak i FIFO. Otwarzamy plik z liczbami i wpisujęmy ich do naszej listy z liczbami, tak samo wypełniamy naszą roboczą listę, tak samo sprawdzamy czy kolejna liczba obecnie znajdują się w liście roboczej, jeżeli tak to plusujemy 1 do gólnej liczby hitów. Między tym my mamy „historie” tych liczb, czyli dodajemy wszytkie liczby po kolei do listy historii,żeby było łatwo zrozumieć ktora liczba była najdawniej używana. Jezeli liczba nie znajduję się w roboczej liście to zaczynamy po kolei sprawdzać historie jeśli ta bieżaca liczba z historii znajduję się w roboczej liście to sprawdzamy czy nie ma tej liczby w tempie(ta lista temp służy dla tego żeby wyznaczyć najdawnią używaną liczbe), to zapisujęmy jej do tej listy. Dalej my już wiemy że otatnia dodana liczba jest najdawnią używaną, temu dodajemy tę liczbę do zminnej i usuwamy jej z listy roboczej i zapisujemy liczbę ktorą iterowaliśmy na początku. Na koniec wyznaczamy hitrate i missrate dzieląc ilość hitów i missów na długość naszej listy z liczbami.

**Opracowane wyniki Eksperymentów**

**FCFS:**

Całkowity czas pracy wszystkich procesów wynosi: 11450

Całkowity czas ociekiwania wynosi: 10968

Średni czas pracy wszystkich procesów wynosi:

Średni czas ociekiwania wynosi:

**SJF:**

Całkowity czas pracy wszystkich procesów wynosi: 7349

Całkowity czas ociekiwania wynosi: 6867

Średni czas pracy wszystkich procesów wynosi:

Średni czas ociekiwania wynosi:

Różnica pomiędzy **FCFS** i **SJF** czasu pracy wszystkich procesów wynosi:

Różnica pomiędzy **FCFS** i **SJF** czasu ocikiwania wynosi:

**FIFO:**

Ilość hitów wynosi: 7

Ilość missów wynosi: 43

Średnia ilość hitów wynosi:

Średnia ilość missów wynosi:

**LRU:**

Ilość hitów wynosi: 8

Ilość missów wynosi: 42

Średnia ilość hitów wynosi:

Średnia ilość missów wynosi:

Różnica pomiędzy **FIFO** i **SJF** ilości hitów wynosi:

Różnica pomiędzy **FIFO** i **SJF** ilości missów wynosi:

**Wnioski**

Analizując wyniki oblicznia możemy stwierdzić że algorytm SJF jest znacznie szybrzy niż FCFS. Widać to w dużych różnicach między średnimi czsami oczekiwania i pracy wszystkich procesów(43,38% ; 46,03%). Co do FIFO i LRU nie widzimy dużej różnicy pomiędzy nimi (13,3% ; 2,35%) ale można stwirdzić że LRU jest szybrzy od FIFO widać to w różnice hitów(13,3%). Można też powiedzieć, że nie jest

to wynik przypadkowy, ponieważ testy przeprowadzono na bardzo dużej liczbie

projektów.