Sprawozdanie dla programu symulacyjnego, testującego algorytmy szeregujące

Bartosz Sochacki

Wstęp:

Wykorzystane w symulacji algorytmy:

- FCFS (First Came First Served)
- LCFS (Last Came First Served)
- SJF (Shortest Job First)
- ROUND ROBIN dla kwantów = {0.5,1,1.5}
    - Kolejka FCFS
    - Kolejka LCFS

Wykorzystane języki i biblioteki:

- Skrypt tworzący czasy wykonywania się procesów:
    - Bash
- Program symulacyjny:
    - Python
    - Numpy (biblioteka odpowiedzialna za wspomaganie obliczeń)

Założenia:

- Wszystkie procesy przychodzą jednocześnie
- Procesy nie przechodzą w stan uśpienia
- Czasy wykonywania się procesów są losowo brane z zakresu liczb (1,20>
- Wykorzystywany jest tylko jeden procesor

Dodatkowe pliki:

- input.txt - plik z czasami procesów poddanymi do testowania
- output.txt - plik z surowymi wynikami eksperymentu

Opis procedury testowania algorytmów:

Do testowania algorytmów wykorzystany został język Python z pakietem Numpy który udostępnia tablice wielowymiarowe oraz dużo funkcji wspomagających obliczeniach, wspomaga ona obliczenia naukowe, działanie na macierzach, obliczenia numeryczne, oraz Bash do stworzenia skryptu generującego 10000 procesów. Pakiet Numpy pozwolił logicznie podzielić czasy procesów wprowadzone z zewnętrznego pliku, w tablicy wielowymiarowej w której każdy rząd jest osobnym ciągiem procesów, tak więc uzyskałem tablice dwuwymiarowa, która ma 100 kolumn oraz 100 rzędów.

Program symulacyjny napisany jest obiektowo, powoduje to większą czytelność kodu, jego lepszą organizacje oraz co najważniejsze można wielokrotnie wykorzystać niektóre z metod za pomocą kompozycji. W programie symulacyjnych, zaimplementowane są 4 różne rodzaje algorytmów, FCFS, LCFS, SJF oraz Round Robin, z uwzględnieniem kolejności First Came First Served oraz Last Came First Served dla odpowiednich kwantów czasu: {0.5,1,1.5}

Utworzone zostały więc 4 klasy odpowiedzialne za implementacje algorytmu, obliczenie czasów oczekiwania/przetwarzania dla wszystkich procesów, następnie na podstawie wyników algorytmów, obliczane są średnie czasy dla każdego z ciągów. Mając już te dane można obliczyć finalny wynik czyli średnią średnich wszystkich 100 ciągów dla każdego algorytmu i dla każdego czasu (przetwarzania lub oczekiwania). Za wyliczenie odpowiednich średnich jest odpowiedzialna instancja klasy Averanges, która zawiera odpowiednie metody, potrzebne do wykonania odpowiednich obliczeń. Wszystkie obiekty odpowiedzialne za implementacje algorytmów tworzone są w klasie main, która zajmuje się wczytaniem odpowiednich danych, wywołanie odpowiednich metod instancji w celu obliczenia odpowiednich czasów/średnich oraz zapisaniem wyników w surowym pliku wyjściowym.

Opracowanie wyniki eksperymentu:

Testowaniu zostało poddane 10000 procesów podzielonych na 100 ciągów. Uzyskanych zostało więc 100 niezależnych pomiarów. Czasy jakie procesy potrzebowały by się wykonać były wybierane losowo z zakresu (1,20>[ms].

Poniższa tabela prezentuje średnie wyniki czasów oczekiwania i przetwarzania dla poszczególnych algorytmów. Wyniki te zostały uśrednione.

Czas oczekiwania jest to czas jaki proces spędził w oczekiwaniu na przydział do procesora.

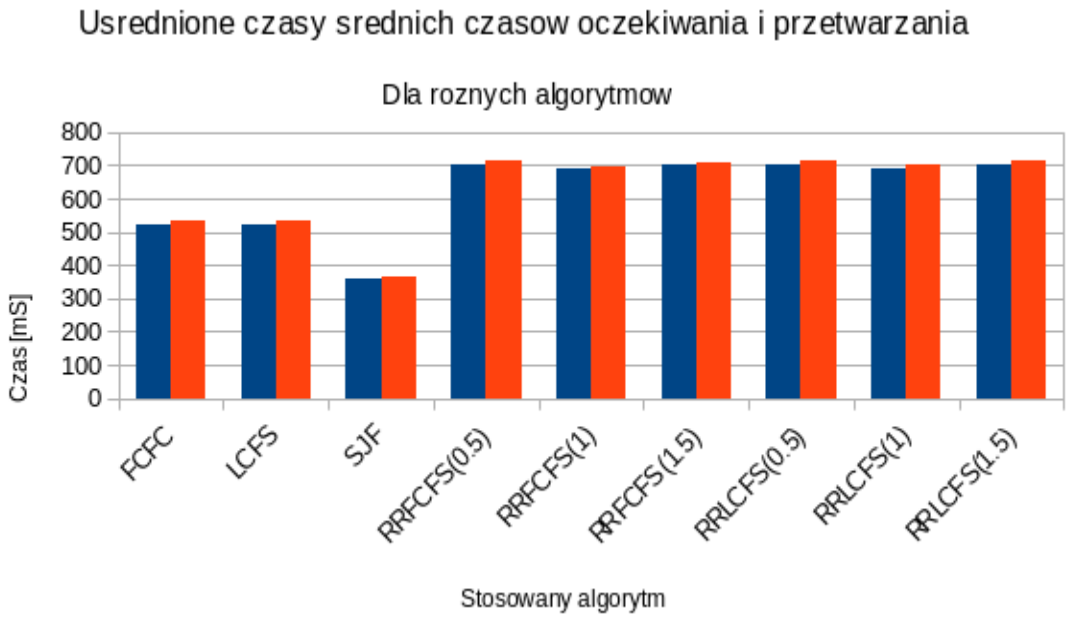
Czas przetwarzania jest to czas jaki proces spędził w systemie, od jego utworzenia aż do zakończenia ostatniej instrukcji maszynowej.

Algorytm:	Średnie czasy oczekiwania:	Średnie czasy przetwarzania:
FCFC	524	535
LCFS	529	540
SJF	363	374
RoundRobinFCFS(0.5)	714.66	725.32
RoundRrobinFCFS(1)	701.54	712.2
RoundRobinFCFS(1.5)	713.34	724.0
RoundRobinLCFS(0.5)	714.91	725.57
RoundRobinLCFS(1)	702.04	712.69
RoundRobinLCFS(1.5)	714.1	724.75

Poniższy wykres prezentuje powyższą tabelę.

-Niebieskie słupki - czasy oczekiwania

-Czerwone słupki - czasy przetwarzania



Wnioski:

Na podstawie powyższych wyników można utworzyć charakterystykę algorytmów które zostały poddane testowaniu i wyciągnąć z niej wnioski.

Algorytm FCFS

Jest to najprostszy w działaniu i implementacji algorytm. Proces, który uzyska dostęp do procesora będzie się wykonywał do końca, nie zostanie on wywłaszczony nawet na rzecz procesu, który na wykonanie potrzebuje bardzo mało czasu procesora. Ponadto algorytm ten jest narażony na sytuacje konwoju, oznacza to, że proces o długim czasie potrzebnym na wykonanie może blokować procesy które na wykonanie potrzebują dużo mniej czasu a co za tym idzie zwiększać średni wynik czasu oczekiwania jak i przetwarzania. Uszeregowanie procesów (czasów jakie procesy potrzebują na wykonanie) w kolejności rosnącej spowoduje pozbycie się efektu konwoju i znaczącego zmniejszenia się średniego czasu oczekiwania i przetwarzania (SJF). Tak więc algorytm ten nie jest optymalnym algorytmem, jeśli zależy nam minimalizacji czasu oczekiwania i przetwarzania

Algorytm LCFS

Jest to algorytm bardzo zbliżony do algorytmu FCFS pod względem średniego czasu oczekiwania jak i przetwarzania z tą różnicą, że LCFS jako pierwszy do procesora dopuszcza proces, który jako ostatni pojawił się w systemie. O tym który czas przetwarzania będzie mniejszy decyduje tak naprawdę losowość. Tak więc dla losowych czasów przydziału procesów do procesora nie można stwierdzić który z algorytmów jest bardziej wydajny.

Algorytm SJF (nie wywłaszczeniowy):

SJF tak jak LCFS jest również w działaniu bardzo podobny do algorytmu FCFS z tą różnicą, że eliminuje on całkowicie sytuację konwoju (dla procesów które przyszły w tym samym czasie). Eliminację sytuacji konwoju uzyskujemy poprzez rosnące uszeregowanie procesów ze względu na czas jaki procesy potrzebują na wykonanie. Jak widać na wykresie i w zamieszczonej wyżej tabeli, sprawia to, że algorytm ten jest najwydajniejszy pod względem czasu oczekiwania i przetwarzania procesu. Średni czas oczekiwania i przetwarzania zmniejsza się, przy użyciu algorytmu SJF, prawie o 200ms względem średnich czasów uzyskanych przez algorytmy FCFS i LCFS. Jest to optymalny algorytm do użycia dla procesów wsadowych w których nie jest istotna szybka reakcja systemu, lecz wydajność wykonywanej operacji w tle.

Algorytm Round Robin:

Charakteryzuje się dużym czasem oczekiwania i przetwarzania w stosunku do pozostałych testowanych algorytmów. Jest to algorytm wywłaszczeniowy. Dopuszcza on proces do procesora na okres odgórnie ustalonego kwantu czasu a następnie wywłaszcza on proces i przydziela on następny proces znajdujący się w kolejce do procesora. Kolejka, zastosowania w symulacji w algorytmie Round Robin to LCFS i FCFS. Wyniki są wręcz identyczne dla obu przypadków. Podtrzymuje to wniosek wywiedziony powyżej, czas oczekiwania i przetwarzania dla kolejki FCFS i LCFS jest bardzo zbliżony. Różnica, w algorytmie Round Robin, pojawia się natomiast dla różnych kwantów czasu, stosowanych przez algorytm. Mianowicie, im mniejszy kwant czasu tym czas oczekiwania i przetwarzania procesu będzie większy. Dzieje się tak ponieważ procesy zostają częściej wywłaszczane, przez co więcej czasu spędzają czekając na przydział do procesora. Dla kwantu czasu 0.5 ms czas oczekiwania jest większy niż dla kwantu czasu równego 1ms. Dla kwantu czasu 1.5 ms czas oczekiwania jest jednak większy niż dla czasu równego 1ms. Spowodowane jest to tym, że procesy, których czasy nie będą wielokrotnością 3 ulegną podzieleniu przez co marnowany jest czas na obsłużenie ułamków czasów procesów. Istotne jest więc to by kwant czasu był liczbą naturalną.

Nie można użyć zbyt dużego czasu oczekiwania, gdyż spowoduje to sprowadzenie algorytmu Round Robin do zwykłego FCFS i co za tym idzie małą responsywność systemu.

Przy implementacji tego algorytmu należy pamiętać by nie dać zbyt małego kwantu procesu, gdyż sama procedura wymiany procesu na inny może być dłuższa niż jego rzeczywista praca. Tak więc dobór kwantu czasu dla algorytmu wydaje się mieć kluczowe znaczenie.

Algorytm Round Robin jest to algorytm, który jest optymalny dla procesów interaktywnych. Zapewnia on szybką reakcję systemu, jednak nie jest on efektywny przy procesach wsadowych, spowalnia on ich rzeczywistą prace.